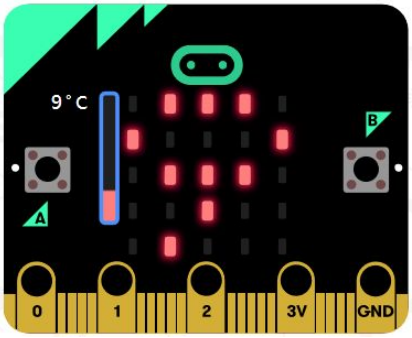
`

Overview

This is an introduction to coding and computer science by way of making and design, using the revolutionary new micro:bit* microcontroller board, and Microsoft's easy and powerful MakeCode block-based coding environment. It is a project-based curriculum with a maker philosophy at its core; the idea is that by making physical objects, students create a context to learn computer science concepts, to think creatively, to code, and to become innovative.



Module 1: Design & Making with Microbit

This module introduces the micro:bit as a piece of hardware that has a specific size and weight, and generally must be supported and incorporated as an essential component of a tangible artifact. Focus on making a pet or robot and incorporating the physical micro:bit as the face of the project.

Module 2: Software & Hardware (Algorithms)

This module introduces a conceptual framework for thinking of a computing device as something that uses code to process one or more inputs and send them to an output(s). Questions to be answered include: What is a computer? What is a microbit and what can it do? Students will be introduced to the basic algorithm of input, processing, and output. Students will be making projects that utilize the microbit sensors to get input, the microbit to process the input, and the LED screen to output the results.

Module 3: Everything Counts (Variables)

Computer programs process information. Some of the information that is input, stored, and used in a computer program as values that vary or change during the running of a program. Programmers create variables to hold the value of information that may change. In a game program, a variable may be created to hold the player's current score, since that value would hopefully change during the course of the game. Students will be making projects that utilize variables, like: a people counter, pedometer, score keeper, and/or dice roll.

Module 4: Making Decisions (Conditionals)

Computer programs are instructions telling the computer how to process input and deliver output. An important part of programming is telling the computer WHEN to perform a certain task. For this, we use something called 'conditionals'. Conditionals get their name because a certain Condition or Rule has to be met. Conditionals are usually implemented using an 'if (condition) then action'

statement. Students will be creating and making projects like coin toss, Magic 8 Ball, and/or dice toss with dots instead of numbers.

Module 5: Music, Designs & LEDs (Loops)

The microbit has several General Purpose Input Output pins (GPIO) that can be used get additional inputs or to direct output to. The ability to process inputs and control external outputs devices is the basis for much of the technology innovation that is happening in the world today. One of the things that computer are really good at is doing the same thing over and over again without getting tired. In this module students will use loops to compose music, connect headphones/speaker, connect LED's, and designs that repeat.

Module 6: Radio Communications

This module covers the use of more than one micro:bit to share and combine data. Students will explore the Radio functionality of the micro:bit. Students will send and receive numbers and strings in a series of guided activities. Finally, students are asked to collaborate so that they can share their micro:bits and create a project together that uses the radios as part of their project.

Module 7: Innovative Project

In this module, students will be reviewing the concepts they covered in the previous weeks, and providing some ideas for an independent "mini-project" students can focus on. We will also introduce a framework for keeping students accountable to the work they are doing individually and in groups, and providing a rubric for assessment of the development process, as well as the finished product.

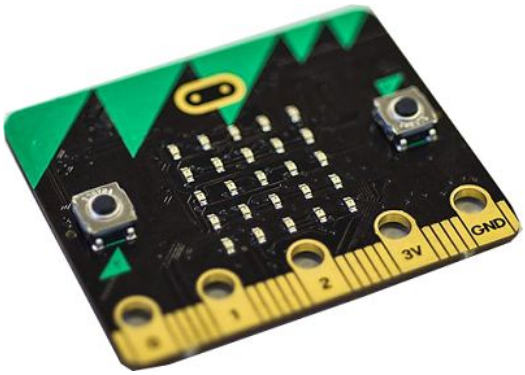
It is important to allow students to practice accounting for the work they are doing on a short "mini-project" like this, so that when they move on to an independent project spanning multiple weeks, it will be easier for you to keep track of what everybody is doing. Programming is a process of patient problem-solving, and finding ways to value, acknowledge, and reward the problem-solving process is an important part of assessment.

References:

Instructions to update the firmware on the Microbit and how to use the MakeCode beta editor with the updated serial connection in the editor.
https://support.microbit.org/support/solutions/articles/19000019131-how-to-upgrade-the-firmware-on-the-micro-bit

01 Design & Making with Micro:bit

This lesson introduces the micro:bit as a piece of hardware that has a specific size and weight, and generally must be supported and incorporated as an essential component of a tangible artifact. Focus on incorporating the physical micro:bit into a basic making activity.



Lesson objectives

Students will…

- Exercise creativity and resourcefulness by coming up with ideas for using simple household materials to accommodate the micro:bit's size and weight in many different ways.
- Test and iterate using different materials and sizes in order to create an optimal design to house the micro:bit and battery pack
- Learn how to download programs and move them to the micro:bit file to run on the micro:bit.
- Use the design thinking process to develop an understanding for a problem or user need.
- Apply their understanding in a creative way by making a "micro:pet" creature.

Lesson plan

- **Introduction**: The micro:bit is for making
- **Unplugged**: Design Thinking
- **Activity**: MakeCode download
- **Project**: micro:pet or micro:robot (including mods and rubric)

Standards — CSTA K-12 Computer Science Standards

- 2-A-2-1 Solicit and integrate peer feedback as appropriate to develop or refine a program.
- 2-A-6-10 Use an iterative design process (e.g., define the problem, generate ideas, build, test, and improve solutions) to solve problems, both independently and collaboratively.

01.0 Topic Introduction

The micro:bit is a great way to teach the basics of programming and computer science. The Microsoft MakeCode block-based coding environment is a powerful and intuitive way to make the micro:bit react to all sorts of input, and you can introduce fundamental concepts such as loops, conditional statements, and variables using MakeCode.

Students often focus primarily on the 5x5 LED screen for providing output. Although this is the most directly accessible way to see a reaction to some kind of input, there are many more creative

possibilities when you encourage your students to see the micro:bit as a “brain” that can control physical, tangible creations.

These creations don’t have to be complex or highly technical. It’s great to have students building with common household supplies. Because the micro:bit is so lightweight, and supports so many sensors, it can be incorporated easily into a physical design as long as students plan ahead for its size and weight. One of the first questions you might ask students is “Where does the micro:bit fit in your creation?”

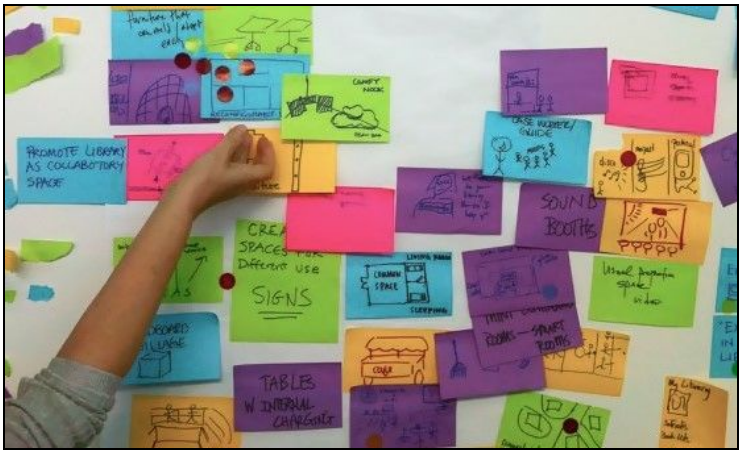
In this first lesson’s project, we focus on making something creative that features the micro:bit as its “face”. We purposely start this course with a lesson on Making and the physical nature of the micro:bit, because it is important to set the tone for the whole course that this is a class about making, building, crafting and construction. It helps if you have an art room available where kids can work, or arts and crafts supplies in your classroom that kids can use to build.

Some common making supplies to gather:

- pizza boxes
- scrap cardboard
- colored construction paper
- colored duct tape
- scissors
- pipe cleaners
- stickers
- feathers
- string
- markers

01.1 Unplugged: Design Thinking

Objective: To introduce a process of design that starts with talking to one another. Whatever you build with code should serve a purpose or fill a need. Sometimes what you build will make the world more beautiful, or help somebody else. Our design process, based on a process called **design thinking**, can give students a specific framework for thinking purposefully about design.



Overview: In this activity, students will interview each other about their ideal pet or robot. They should take notes. The first step in

coding by design involves understanding someone else's need. Then, you can create prototypes that get you closer and closer to the best solution.

Materials: Pairs of students, something to take notes on.

Getting started: Pair students up with each other. One is Student A, the other is Student B. The goal of this activity is to gather information from their partner that will help them to design a micro:bit pet or robot for their partner.

5 minutes: Student A interviews Student B. The goal is to find out what Student B considers to be their ideal pet or robot. Student A should mostly listen, and ask questions to keep Student B talking for the entire time. Student A makes some notes in Student B's notebook. Here are some questions to start with:

- What would you like to have a pet or robot? What is it?
- What would you like about pet or robot? What do you dislike?
- Is there anything you wish your pet or robot could do? Why?
- Tell me about your ideal pet or robot.

5 minutes: Student B interviews Student A, as above.

The goal is to find out more about your partner by asking questions. Try to ask "Why?" as much as possible. Your partner will tell you about his or her ideal pet or robot, but you are really finding out more about your partner's likes and dislikes. When we design, we create real things for real people. So we need to start with understanding them first.

5 minutes: Student A and Student B review their notes, and circle anything that seems as if it will be important to understanding how to create the ideal pet or robot for themselves. Circle ideas, advice, anything that could be helpful when they start building. Then, they should use what they have discovered with their partner to fill in the blanks:

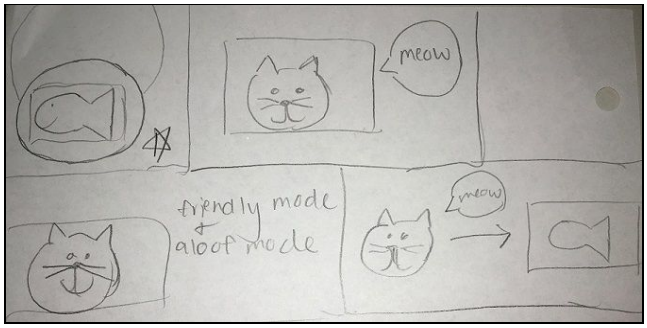
"My partner needs a _____ because _____."

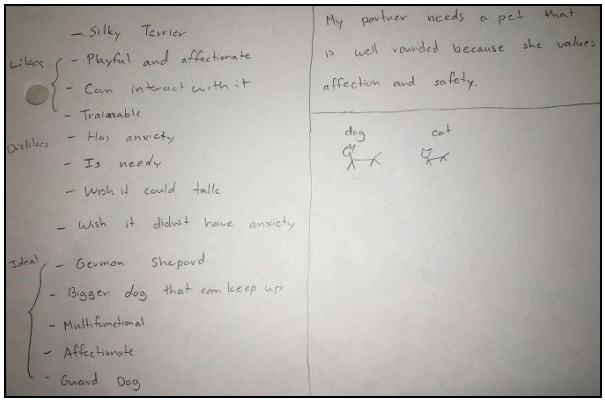
This definition statement should draw some conclusions about their partner's need based on the conversation they have had with that person.

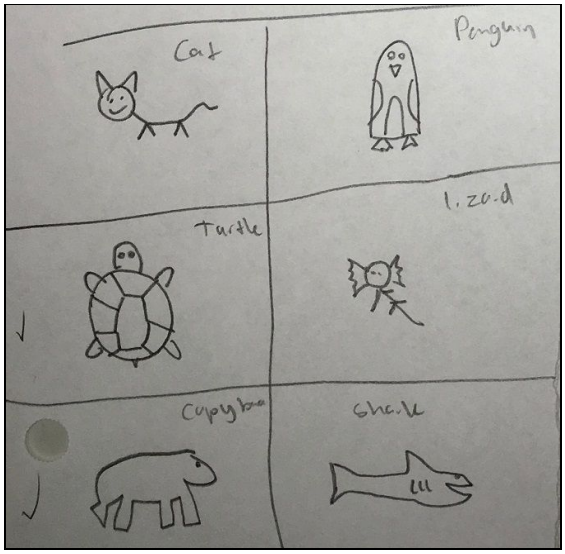
5 minutes: Student A and Student B sketch at least 2-4 ideas of pets or robots from the questions their partner asked. Stick figures and diagrams are okay. At this point, quantity is more important than quality. Students shouldn't limit themselves to real animals or robots; fictional pets or far out robots and mashups are totally fine!

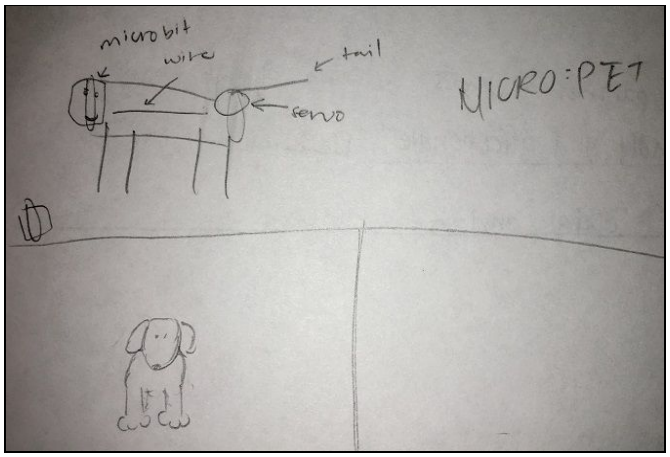
Make sure students keep their notes and sketches! They will use them in the project for this lesson.

Examples









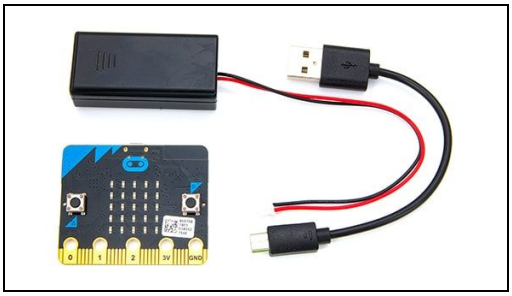
01.2 Activity: Installing a program

micro:bit activity: Installing a Microsoft MakeCode Program on the micro:bit

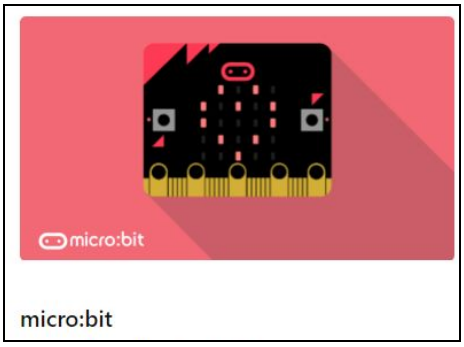
Objective: Learn how to download programs from the MakeCode tool.

Overview: Students will create a simple program in **Microsoft MakeCode** and download it to their micro:bit using a USB cable.

For this activity, students will each need a micro:bit, a micro-USB cable, computer, and a battery pack.



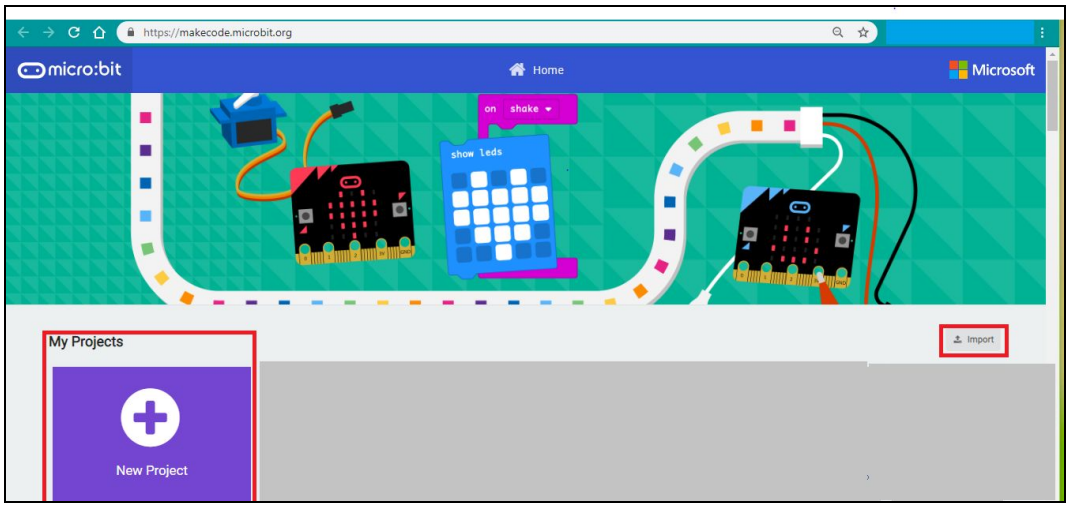
a



Open a browser window to makecode.com, and select the micro:bit code editor.

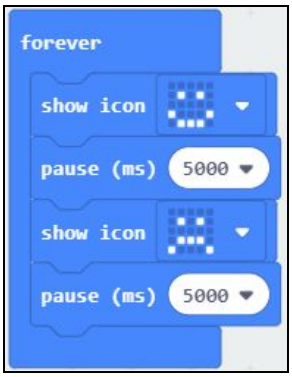
MakeCode Microbit Home screen

From the MakeCode.Microbit.org Home screen, select the **"+" New Project** from the **My Projects** menu to start a new project. To load a previous project, click on the file name in the **My Projects** menu. To load a previous project from a compiled MakeCode.Microbit "*.hex" file click on **Import File** and browse to find the file. Select the file that you saved on your computer in the previous step.



The program should look like the following in MakeCode. It shows a repeating series of faces:

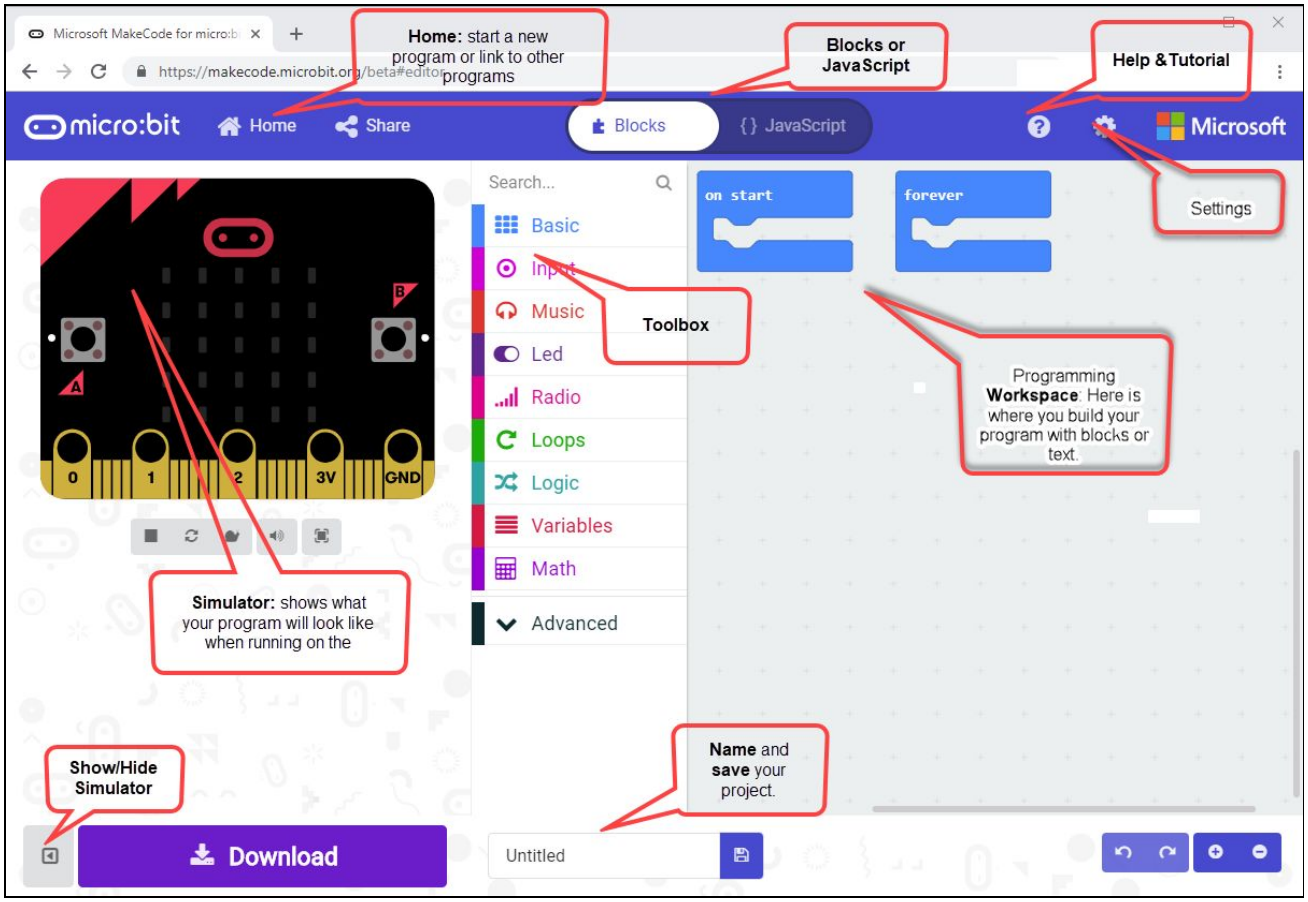
MakeCode blocks version	***JavaScript version***

	`basic.forever(() => {`
	`  basic.showIcon(IconNames.Happy)`
	`  basic.pause(5000)`
	`  basic.showIcon(IconNames.Sad)`
	`  basic.pause(5000)`
	`})`

Tour of Microsoft MakeCode

- **Simulator** - on the left side of the screen, you will see a virtual micro:bit that will show what your program will look like running on a micro:bit. This is helpful for debugging, and instant feedback on program execution.

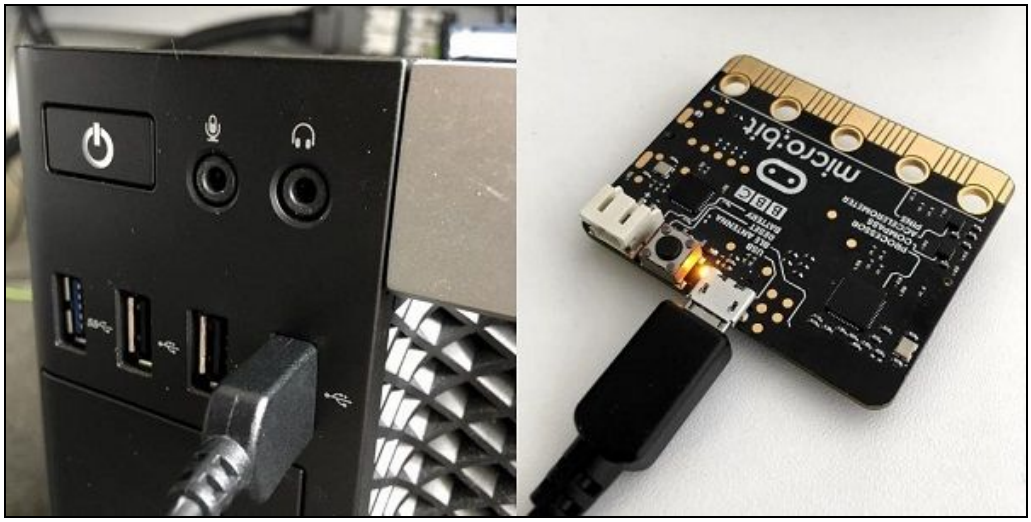
- **Toolbox** - in the middle of the screen, there are a number of different categories, each containing a number of blocks that can be dragged into the programming workspace on the right.
- **Workspace** - on the right side of the screen is the Programming Workspace where you will create your program. Programs are constructed by snapping blocks together in this area.



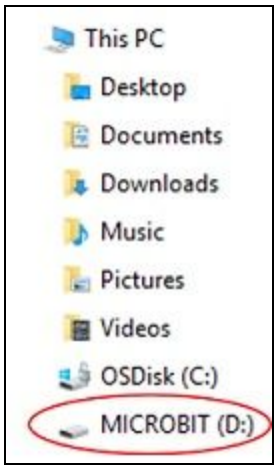
The color of the blocks identifies their category. All of the blocks that make up the program above come from the **Basic** Toolbox category, which is light blue.

Downloading a MakeCode program to the micro:bit

To download the file to your micro:bit, you must connect it to your computer's USB port using a micro-USB cable. The micro:bit will draw power from your computer through the USB connection, or you can connect an optional battery pack so it can function even after it is unplugged from the computer. Once plugged in, the micro:bit shows up on your computer like a USB flash drive.

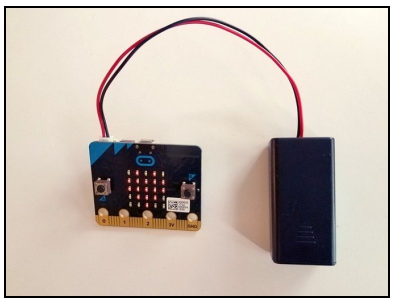


Click the purple Download button in the lower left of the MakeCode screen. This will download the file to your computer, to the location where your browser is set to save downloads.



To move the program to your micro:bit, drag the downloaded "**microbit-xxxx.hex**" file to the MICROBIT drive, as if you were copying a file to a flash drive. The program will copy over, and it will begin running on the micro:bit immediately.

The micro:bit will hold one program at a time. It is not necessary to delete files off the micro:bit before you copy another onto the micro:bit; a new file will just replace the old one.



For the next project, your students should attach the battery pack (it takes 2 AAA batteries) to the micro:bit using the white connector. That way they can build it into their design without having to connect it to the computer.

01.3 Innovation Project: Micro:pet or Micro:robot

This project is an opportunity for students to create a micro:pet or micro:robot for the partner they interviewed in the Unplugged activity. They should review their notes and try to summarize what their partner finds appealing in a pet. Then, they should use whatever materials are available to create a prototype of a pet their partner would like.

We often ask students to sketch a few designs on paper first, then consult with their partner to see which aspects of those designs they find most appealing. The purpose of prototyping is to gather more feedback to help you in your final design ("I like this part from Idea A, and I like this part from Idea B…")

Build a micro:pet or micro:robot that:

- Matches your partner's needs
- Supports the micro:bit and its battery pack
- Allows you to easily access the micro:bit to turn it on and off

Your design should use whatever materials are available to support the micro:bit so that its face is showing. You can be creative and decide how to mount the board, and how to decorate your critter.

Think about the following questions when you construct it:

- Will it be an animal? A plant? A robot? A bug?
- Will it have any moving parts?
- If it moves, how can you hold the micro:bit securely?

Some photos of sample micro:pets below!

Ideas for Modifications

- Find a way to make part of the animal or robot move.
- Give your animal or robot a natural habitat.
- Create a way to carry your animal or robot.
- Create an animal or robot that reacts when you pet it or move it (find a way to detect when the micro:bit is moved or when its position changes in a certain way.)

Reflection

Have students write a reflection of about 150–300 words, addressing the following points:

- Summarize the feedback you got from your partner on your idea. How would you revise your design, if you were to go back and create another version?
- What was it like to have someone designing a pet or robot for you? Was it a pet or robot you would have enjoyed? Why or why not? What advice did you give them that might help them redesign?
- What was it like to interview your partner? What was it like to be listened to?
- What was something that was surprising to you about the process of designing the micro:pet?
- Describe a difficult point in the process of designing the micro:pet, and explain how you resolved it.

Reflection handout. https://goo.gl/C4oLWH or “Coding & Innovation using Microbits” booklet http://bit.ly/codingmicrobitsbooklet

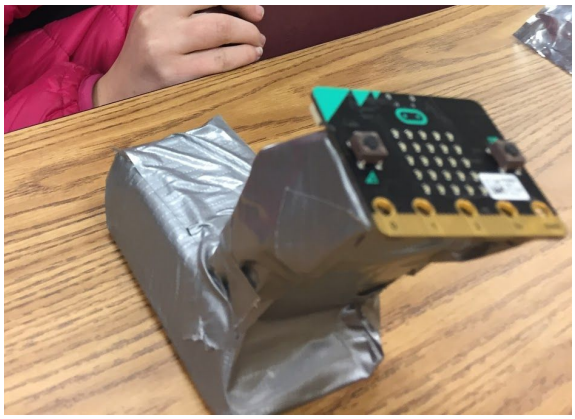
Rubric

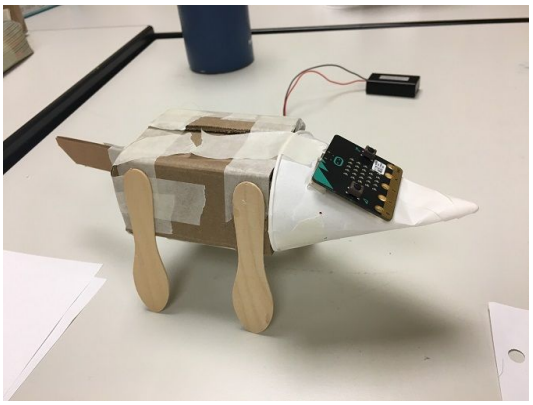
For creative projects such as these, we normally don’t use a qualitative rubric to grade the creativity or the match with their partner’s needs. We just check to make sure that the micro:pet meets the required specifications:

- Program properly downloaded to micro:bit
- micro:bit supported so the face is showing
- micro:bit can be turned on and off without taking critter apart
- Turned in notes on interview process
- Written reflection (prompt is above)

Micro:pet & Micro:Robot Examples

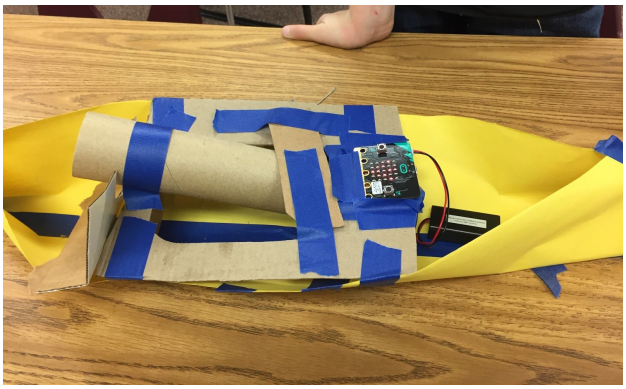
Dogs



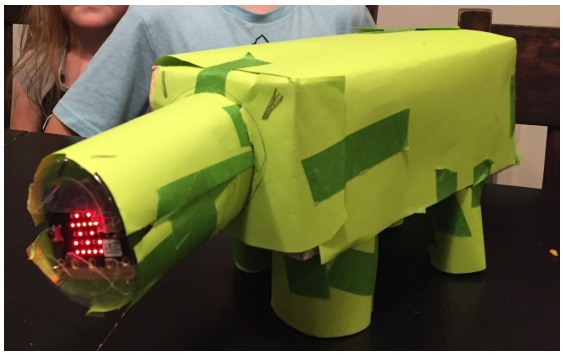


Robot

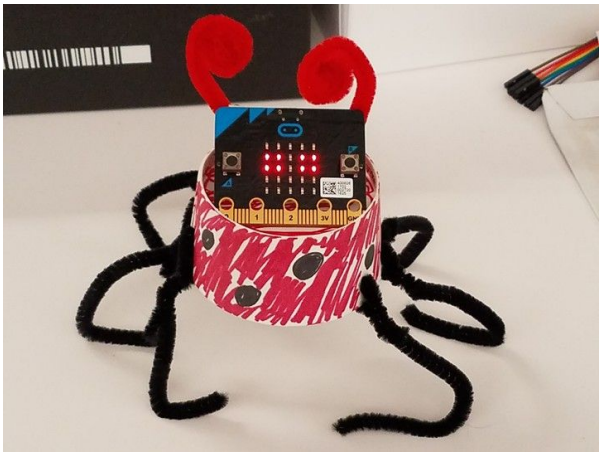
Battleship



Alligator

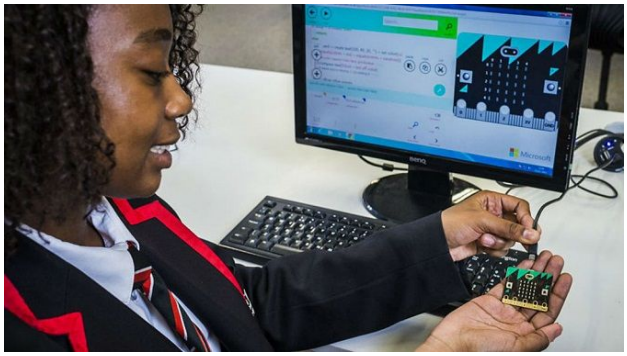


Ladybug



02 Software & Hardware (Algorithms)

This lesson introduces a conceptual framework for thinking of a computing device as something that uses code to process one or more inputs and send them to an output(s).



Lesson objectives

Students will…

- Understand the four components that make up a computer and their functions.
- Understand that the micro:bit takes input, and after processing the input, produces output.
- Learn the variety of different types of information the micro:bit takes in as input.
- Apply this knowledge by creating a micro:bit program that takes input, processes it, and produces an output.

Lesson plan

1. **Overview**: What is a computer and micro:bit hardware
2. **Unplugged**: What's your Blackbox? IPO Blackbox revealed
3. **Activity**: Sensors - Temperature, Compass, etc,
4. **Project**: Blackbox

Standards — CSTA K-12 Computer Science Standards

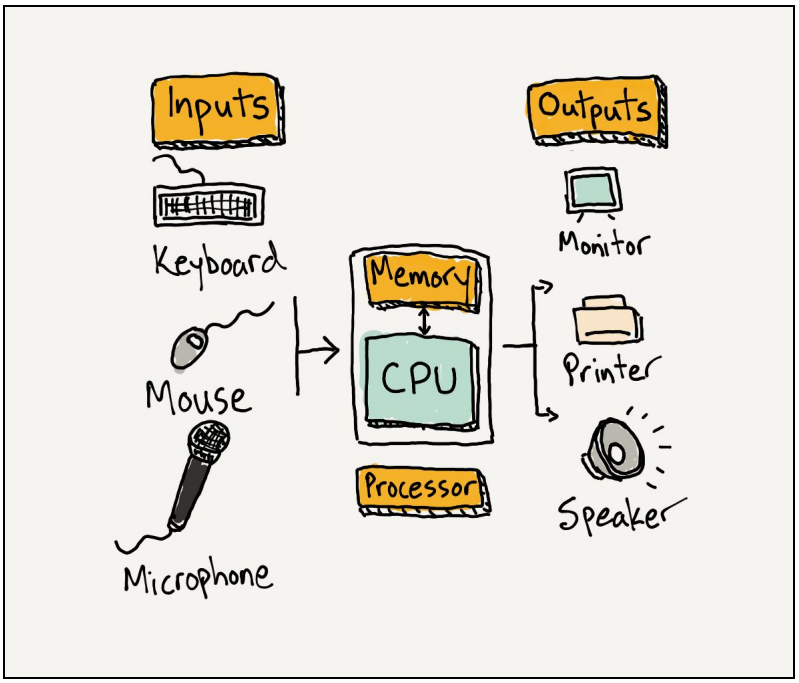
- CT.L2-03 Define an algorithm as a sequence of instructions that can be processed by a computer.
- CD.L2-01 Recognize that computers are devices that execute programs.
- CD.L2-02 Identify a variety of electronic devices that contain computational processors.
- CD.L2-03 Demonstrate an understanding of the relationship between hardware and software.
- CD.L3A-04 Compare various forms of input and output.

02.0 Introduction

What is a micro:bit? The micro:bit was created in 2015 in the UK by the BBC to teach computer science to students. The BBC gave away a micro:bit to every 7th grade student in the UK. You can think of a micro:bit as a mini computer. http://microbit.org

What is a computer? There are 4 main components that make up any computer:

1. The **Processor** – this is usually a small chip inside the computer, and it's how the computer processes and transforms information. Has anyone heard of the term "CPU"? CPU stands for Central Processing Unit. You can think of the processor as the Brains of the computer - the faster the processor, the more quickly the computer can think.
2. The **Memory** – this is how the computer remembers things. There are two types of memory:
    - RAM (random access memory) - you can think of this as the computer's short-term memory
    - Storage (also referred to as the "hard drive") - this is the computer's long-term memory, where it can store information even when power is turned off



3. **Inputs** – this is how a computer takes in information from the world. On humans, our input comes in through our senses, such as our ears and eyes. What are some Computer Inputs? Keyboard, Mouse, Touchscreen, Camera, Microphone, Game Controller, Scanner
4. **Outputs** – this is how a computer displays or communicates information. On humans, we communicate information by using our mouths when we talk. What are some examples of communication that don't involve talking? Blushing, sign language. What are some examples of Computer outputs? Monitor/Screen, Headphones/Speakers, Printer

Computers in your house? How many can you list? Share your list with a partner. Did you think of additional computers that you can add to your list?

Apple Watch

In 2015 Apple computer introduced their first Apple Watch. **Apple Watch** is a line of smartwatches designed, developed, and marketed by Apple Inc. It incorporates fitness tracking and health-oriented capabilities with integration with iOS and other Apple products and services.

(Illustrations iPhone screenshots from Apple Watch data: Active Energy- Move, Exercise, Stand; Sleep; Heart Rate graph and data)

Apple Watch relies on a wirelessly connected iPhone to perform many of its default functions such as calling and texting. However, Wi-Fi chips in all Apple Watch models allow the smartwatch to have limited connectivity features away from the phone anywhere a Wi-Fi network is available. (https://en.wikipedia.org/wiki/Apple_Watch)

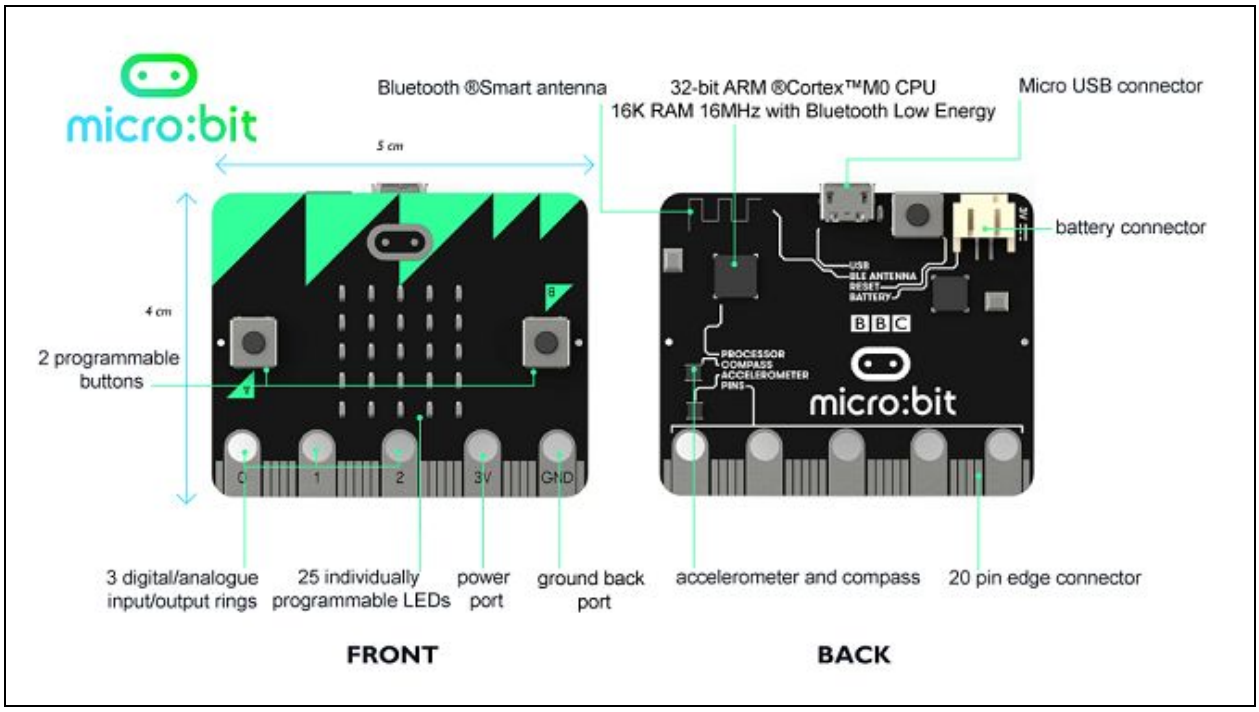
What inputs or sensors are in an Apple Watch?

What outputs does an Apple Watch have?

What can be monitored with an Apple Watch?

Micro:bit

Now, let's look at our micro:bit:



- Use the diagram here as a visual aid: http://microbit.org/hardware/
- Can you find the Processor?
- How much memory does the micro:bit have? 16K, which is smaller than many files on your computer!
- Can you locate the following Inputs? Buttons (on board), Pins (at base), Accelerometer / Compass.
- Note: Though not pictured, the Light Sensor is located on the LED lights
- Where are the Outputs? LED lights, Pins

All computers need electricity to power them. There are 3 ways to power your micro:bit:

- Through the USB port at the top
- By connecting a battery pack to the battery connector
- Through the 3V Pin at the bottom (not the recommended way to power your micro:bit)

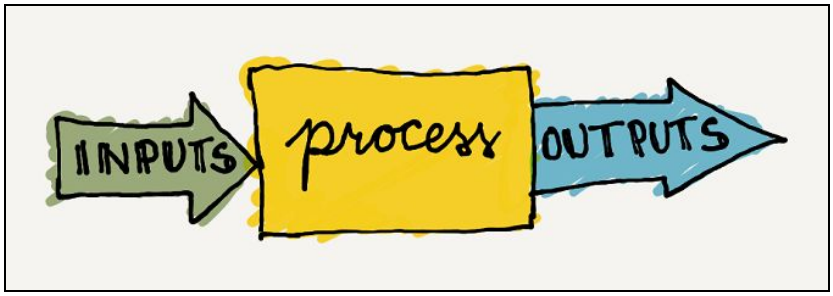
On the top left corner you may notice that your micro:bit has a Bluetooth antenna. This means your micro:bit can communicate and send information to other micro:bits. We will learn more about this feature in the Radio Lesson.

02.1 Unplugged: What's in your Blackbox? IPO - Blackbox revealed.

Materials

- Pencils
- Paper (or index cards)

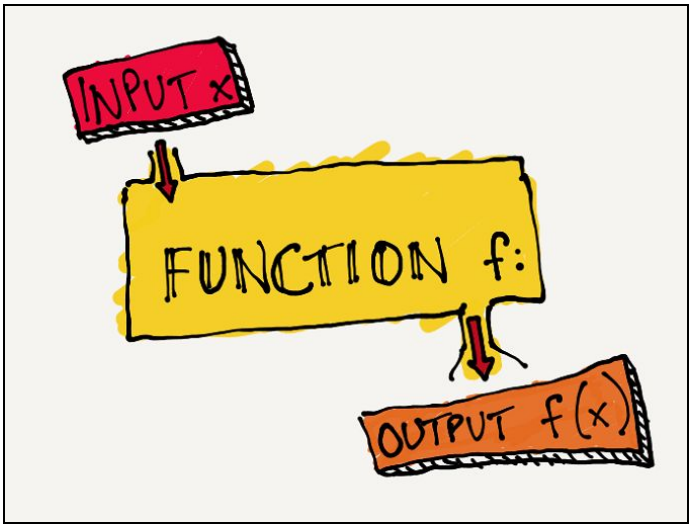
Algorithms



In computer programming, **algorithms** are sets of instructions.

Algorithms 'tell' the computer how to process input and what, if any, output to produce.

Blackbox



"… a **black box** is a device, system or object which can be viewed in terms of its inputs and outputs (or transfer characteristics), without any knowledge of its internal workings. Its implementation is "opaque" (black). Almost anything might be referred to as a black box." (https://en.wikipedia.org/wiki/Black_box)

Many computer programs can be thought of as a blackbox that takes an input, processes it in some way, and outputs the result. If the blackbox is opened up we can see how the input is processed and then output. When creating an algorithm for a computer program is many times helpful to think of it is the terms of **Input, Process**, and **Output (IPO)**.

Function Machine

An example of an algorithm you have seen in math class is the 'function machine'.

A function machine takes an input, processes the input, and then delivers an output. When you can see the process in the function machine you have opened up the blackbox!

The inputs and their outputs are usually recorded in an input output table, where the value of x represents the input and the value of y represents the output. See example.

Input (x)	Output (y)
1	2
2	4
3	6
4	8

A common math problem is to determine what processing is happening to the input that results in the given output. In the example above, each input is being doubled (multiplied by 2) to produce the corresponding output.

Input (x)	Processing =>	Output (y)
1	* 2	2
2	* 2	4
3	* 2	6
4	* 2	8

The **algorithm** for this might be thought of as:

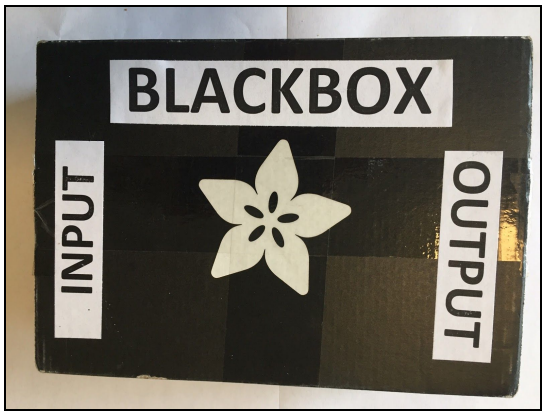
// Input number
Get a number as input

// Process the number
answer gets its value from number * 2

// Output results
display the answer on the screen

Unplugged: What's in your Blackbox?

For this activity, the students can work in pairs, Player A and Player B. The pairs will take turns running the Blackbox machine for their partner who will be providing **Input** to be **processed** and the person running the Blackbox **Outputs** the answer. The person inputting tries to guess what the calculation is the Blackbox is making. (Link to Blackbox labels document: http://goo.gl/jffnZT so students can make their own blackbox.)



Direct the students how you would like them to record their work. They can use pencil and paper or index cards. On paper, they can keep track of inputs and outputs in a table (see example above). With index cards, Player A can write each input on one side of an index card, hand the card to Player B, who passes it into the Blackbox and then writes the corresponding output on the other side of the card and brings it out of the Output side of the Blackbox.

To begin:

- Player B decides on a mathematical function or bit of processing* that will be done on whatever input she receives from Player A.
- Player B should write down the function or bit of processing and set it aside, out of sight of Player A.
- Player A then gives Player B a number to process.
- Player B processes the number and returns an output to Player A.
- Player A can then state what function or bit of processing she thinks Player B is using on the input to produce the given output. One try per round of input/output.
- If Player A states the correct function, Player B confirms that it is correct by showing the previously hidden function and the players switch roles and start the game over.
- If Player A does not guess correctly, Player A provides another input that Player B processes and provides an output for.
- The goal is for Player A to figure out what function or bit of processing Player B is using in the fewest number of rounds of input/output possible.
- After each student has had at least one chance to be the function machine, play more rounds as time permits.

Notes:

- The difficulty level of the possible functions should be determined by the teacher and shared with the students ahead of playing. Alternately, the teacher can provide function cards that are handed out at random to be used by the players, rather than the players creating their own.
- The player providing the input should not just guess what the function is. She should be able to explain why she thinks her input resulted in the given output.
- Examples of ‘easier’ functions:
    - Add 8
    - Subtract 6
    - Multiply by 3
    - Divide by 2
- Examples of more difficult functions:
    - Multiply by 2 and then subtract 1
    - Square the input
    - Return 20% of the input

02.2 Activity: Sensors - Temperature, Compass, etc.

The micro:bit itself is considered hardware. It is a physical computing piece of technology. In order to make use of hardware, Software needs to be written (otherwise known as “code” or computer programs). The software “tells” the hardware what to do, and in what order to do it using algorithms. Algorithms are sets of computer instructions.

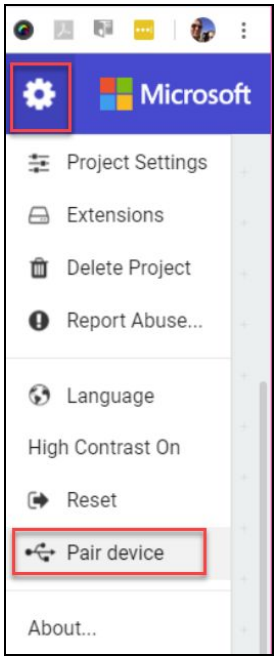
In this activity, we will discover how to use the micro:bit buttons to **Input** data from the Microbit sensors, **Process** the data, and **Output** it to the LED screen. We will also learn about pseudocode, the MakeCode tool, event handlers, and commenting code.

Algorithm & Pseudocode

What do you want your program to do? The first step in writing a computer program is to create a plan for what you want your program to do. Write out a detailed step-by-step plan for your program. Your plan should include what type of information your program will **input**, how this input will be **processed**, what **output** your program will create and how the output will be recorded or presented. Your writing does not need to be written in complete sentences, nor include actual code. This kind of detailed writing is known as pseudocode. Pseudocode is like a detailed outline or rough draft of your program. Pseudocode is a mix of natural language and code.

For the program we will write, the pseudocode might look like this:

- *Start with a screen “SENSORS” display*
- *Press button A*
    - *Input: Get data from the temperature sensor*
    - *Process: Have microbit convert the data into temperature information*
    - *Output: Display the temperature on the LED display*
- *Press button B*
    - *Input: Get data from the compass sensor*
    - *Process: Have microbit convert the data into heading degrees information*
    - *Output: Display the heading on the LED display*



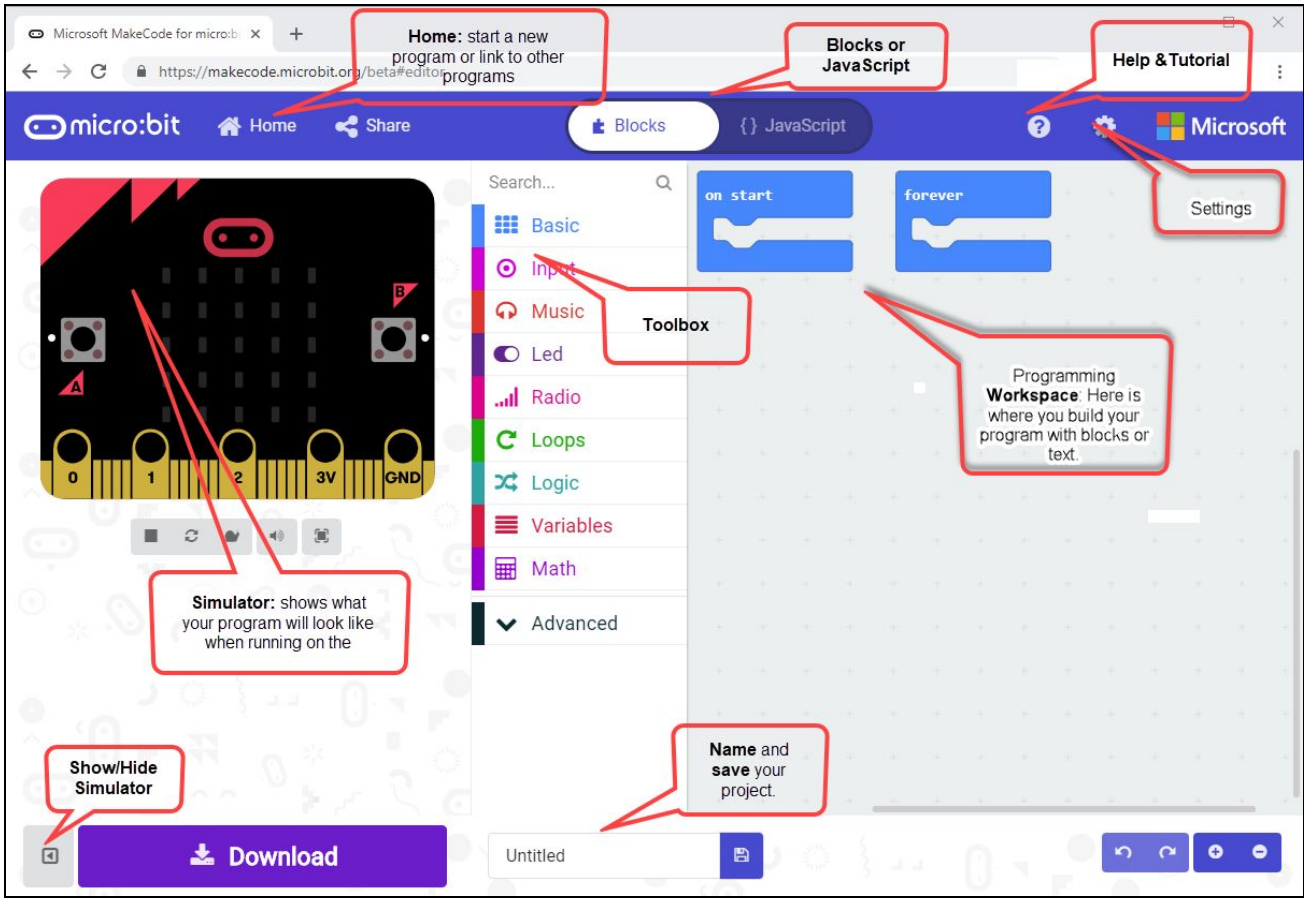
Microsoft MakeCode

Now that you have a plan for your program, in the form of pseudocode, let’s start creating the real program. In a browser window, open the Microsoft MakeCode for micro:bit tool (https://makecode.microbit.org). The MakeCode tool is called an IDE (Integrated Development Environment), and is a software application that contains everything a programmer needs to create, compile, run, test, and even debug a program. All of the screenshots in this section come from version 1 of the MakeCode app as of

(26 October 2018). The version 1 can be accessed at: https://makecode.microbit.org/. The version 1 allows the pairing of the microbot to the computer when the Chrome browser is used by going to the **Settings** gear and selecting **'Pair device'.** When the device is paired, the **Download** button will save directly to the Microbit without having to drag it to the Microbit using a file manager. A paired microbit can also be used to write 'serial" data back to the MakeCode IDE where is can be graphed or downloaded as a CSV file. The firmware of the Microbit may need to be updated for the pairing to work. Here is a link to instructions to update the Microbit firmware. https://support.microbit.org/support/solutions/articles/19000019131-how-to-upgrade-the-firmware-on-the-micro-bit

Tour of Microsoft MakeCode - IDE

- **Simulator** - on the left side of the screen, you will see a virtual micro:bit that will show what your program will look like running on a micro:bit. This is helpful for debugging, and instant feedback on program execution.
- **Toolbox** - in the middle of the screen, there are a number of different categories, each containing a number of blocks that can be dragged into the programming workspace on the right.
- **Workspace** - on the right side of the screen is the Programming Workspace where you will create your program. Programs are constructed by snapping blocks together in this area.



Event handlers

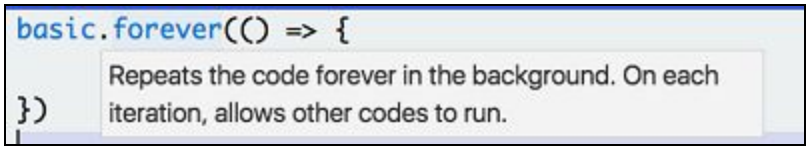
When you start a new project, there will be two blue blocks, ‘on start’ and ‘forever’ already in the coding workspace. These two blocks are event handlers.

In programming, an event is an action done by the user, such as pressing a key or clicking a mouse button. An event handler is a routine that responds to an event. A programmer can write code telling the computer what to do when an event occurs.

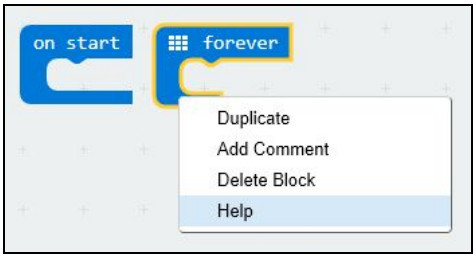
Notes:

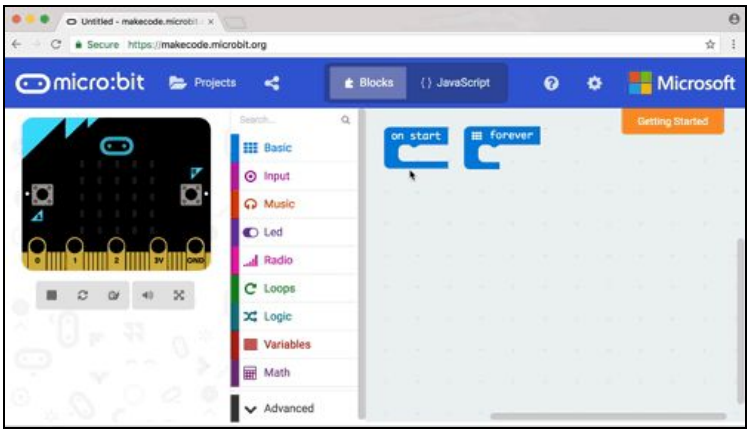
- **Tooltips** - Hover over any block until a hand icon appears and a small text box will pop up telling you what that block does. You can try this now with the ‘on start’ and ‘forever’ blocks. Notice that it also shows you the equivalent code in JavaScript.

Hovering over the code in JavaScript has the same effect.



- **Help/Documentation** - You can also right-click on any block and select Help to open the reference documentation.
- **Deleting blocks** - Click on the ‘forever’ block and drag it left to the Toolbox area. You should see a garbage can icon appear. Let go of the block and it should disappear. You can drag any block back to the Toolbox area to delete it from the coding workspace. You can also remove a block from your coding window by selecting the block and then pressing the “delete” key on your keyboard (or command-X on a Mac).





Name the program

One of the first thing that should be done is to give the program a name in the bottom of the programming environment.

Basic menu

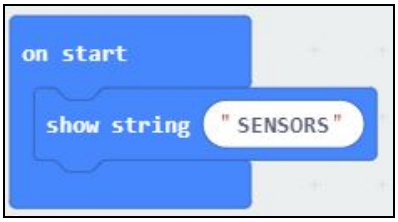
Looking at our pseudocode, we want to make sure to start a program with the name of the program on the microbit, SENSORS.

- We can do this by going to the **Basic menu** -> More and choosing a '**show string'** block.

- Drag the 'show string' block to the coding Workspace.

Notice that the block is 'grayed' out. If you hover over the 'grayed out' block, a pop up text box will appear letting you know that since this block is not attached to an event handler block, it will not run.

- Go ahead and drag the 'show string' block into the 'on start' block. Now the block is no longer grayed out, indicating that it will run when the event, the program starts, occurs. Add the title, "SENSORS" where the word Hello was. All caps makes the text easier to read in the LED display.



MakeCode Simulator

We now have a working program running on the micro:bit simulator! As you write your program, MakeCode will automatically compile and run your code on the simulator. The program doesn't do much at this point, but before we make it more interesting, we should name our program and save it.

On the bottom left of the application window, to the right of the **Download** button, is a text box in which you can name your program. After naming your program, press the **Save** button to save it.

Important: Whenever you write a significant piece of code or just every few minutes, you should save your code. Giving your code a meaningful name will help you find it faster from a list of programs and will let others know what your program does.

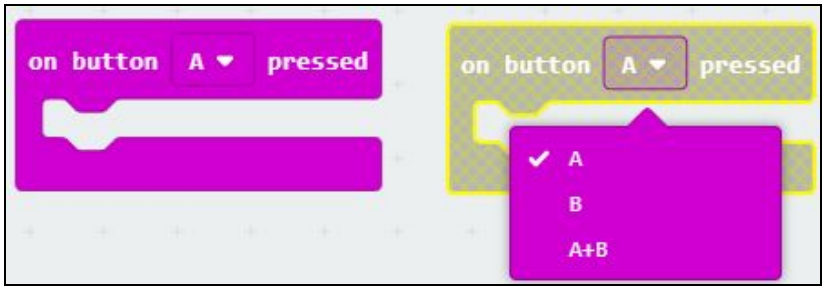
More event handlers

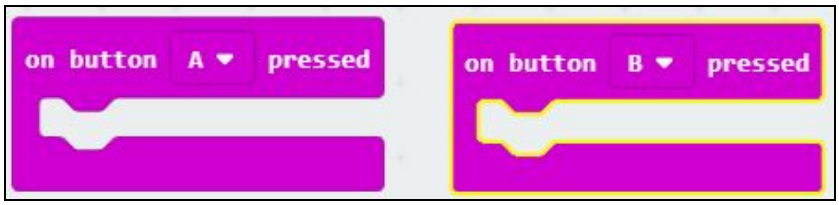
Now to make our program a bit more interesting by adding two more event handlers.

- From the Input menu, drag two ‘on button A pressed’ blocks to the coding window.

Notice that the second block is grayed out. This is because, right now, they are the same block, both ‘listening’ for the same event ‘on button A pressed’.

- Leave the first block alone for now, and using the drop-down menu within the second block, change the ‘A’ to ‘B’. Now this block will no longer be grayed out, as it is now listening for a different event, ‘on button B pressed’.



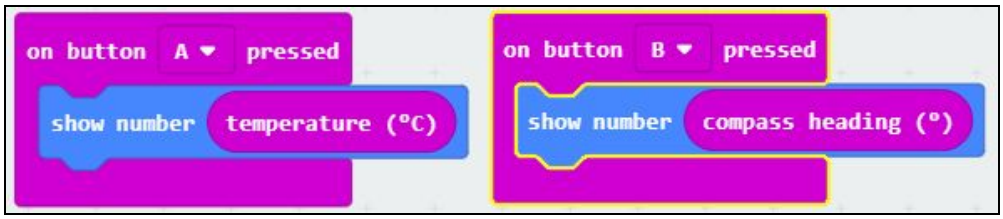


Show Number

Now we can use our **show Number** block to display data from the temperature sensor and output it as a Celsius temperature.

- From the Basic menu, drag two ‘show number’ blocks to the coding workspace
- Place one ‘show Number’ block into the ‘on button A pressed’ event handler and the second ‘show Number’ block into the ‘on button B pressed’ event handler.

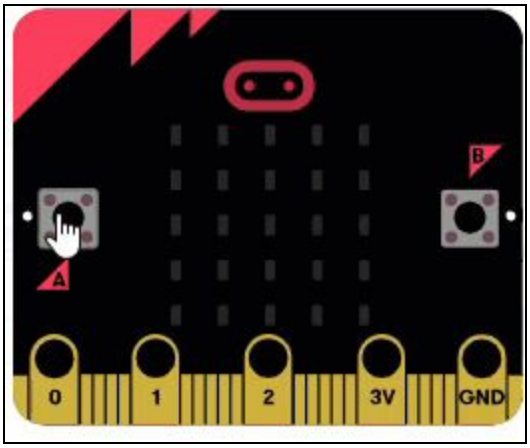
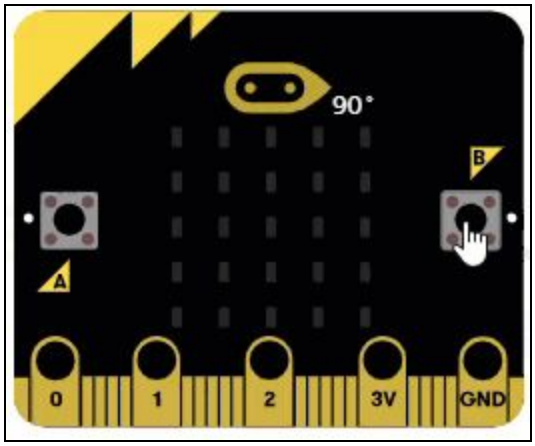
- From the Inputs toolbox locate the ‘temperature’ block and drag it to the workspace and drop it where the ‘0’ is in the input circle. (The outline of the circle will turn yellow when it is in place.)
- From the Inputs toolbox locate the ‘compass heading’ block and drag it to the workspace and drop it where the ‘0’ is in the input circle. (The outline of the circle will turn yellow when it is in place.)



Test your program!

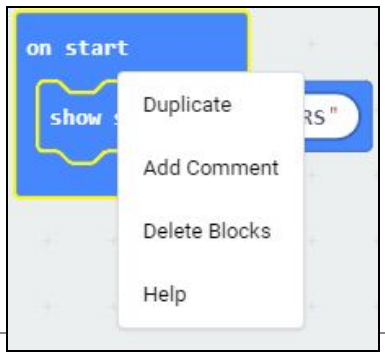
Remember, MakeCode automatically compiles and runs your program, so all you need to do now is press button A to see the temperature and then button B in the simulator to see compass direction produced by your code.

- Feel free to play around with the thermometer and the compass direction in the simulator and show compass heading.
- Remember to save your code.

Temperature in the Simulator	Compass Heading in the Simulator
	

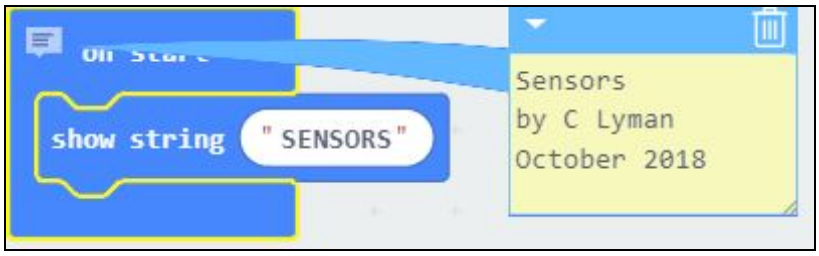
Commenting your code

It is good practice to add comments to your code. Comments can be useful in a number of ways. Comments can help you remember what a certain block of code does and/or why you chose to program something the way you did. Comments also help others reading your code to understand these same things.



To comment a block of code:

- Right-click on the icon that appears before the words on a block.
- A menu will pop up. Select ‘Add Comment’.
- This opens up a bubble dialog for the comment to be typed in. Type in your comment in the dialog bubble. (At the main ‘on Start’ I usually add a program title, my name, and the date it was created.)
- Click on the ‘3 lines’ icon in the code block to close the comment box.
- Click on the ‘3 lines’ icon whenever you want to see your comment again or to edit it.



Notes

- When you right-click on the icon that appears before the words on a block, notice that there are other options available to you that allow you to duplicate and delete blocks, as well as get help. Feel free to explore and use these as you code.
- In **JavaScript**, you can add a comment by using two forward slashes, then typing your comment. The two forward slashes tell JavaScript that the following text (on that same line) is a comment.

// Display the temperature when button A is pressed.

Cleaning up!

Clean up your coding workspace before you do a final save! What does this mean?

- It means that only the code and blocks that you are using in your program are still in the workspace.
- Remove (delete) any other blocks that you may have dragged into the coding workspace as you were experimenting and building your program.

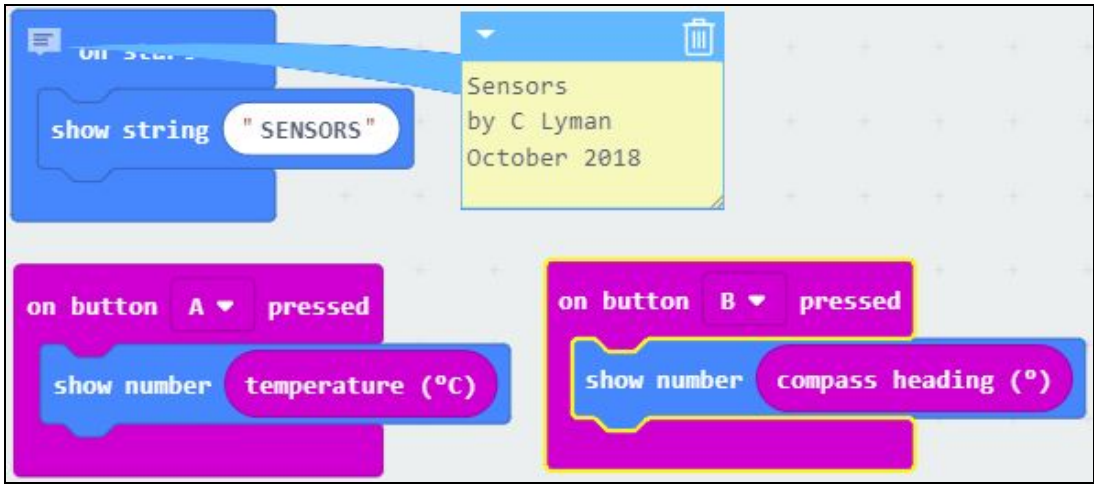
Save and download

Now that your code is running just fine in the simulator, is commented, and your coding window is ‘clean’, save your program, download it to your micro:bit, and enjoy!

Sensors Code

Here is the complete block program:

MakeCode blocks:



JavaScript:

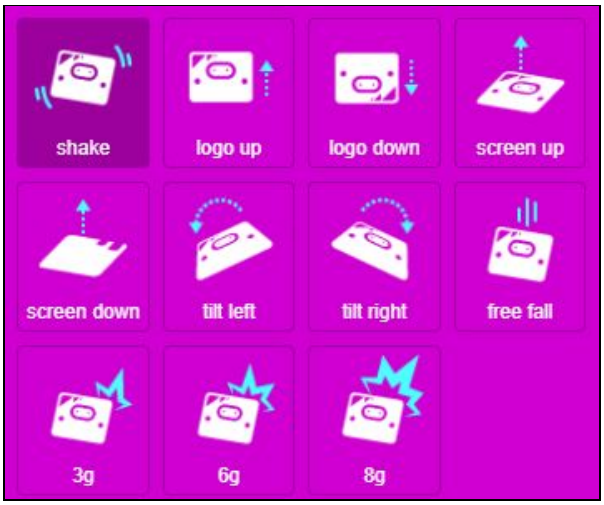
```
// Sensors Program
// by C Lyman
// October 2018
input.onButtonPressed(Button.A, function () {
    basic.showNumber(input.temperature())
})
input.onButtonPressed(Button.B, function () {
    basic.showNumber(input.compassHeading())
})
basic.showString("SENSORS")
```

Modifications

- Try and use some different sensors
- Add an “A+B” button event with a sensor in it
- Figure out how to calculate and display temperature in Fahrenheit degrees

02.3 Innovation Project: Blackbox

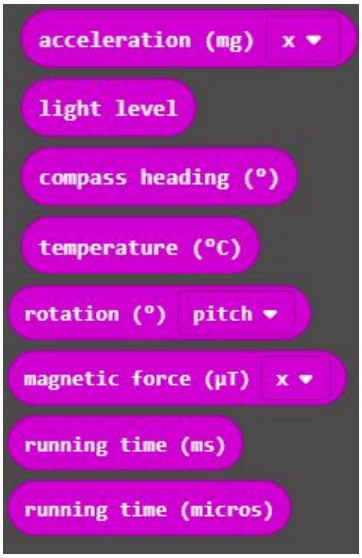
In this project students will explore other events to get input from other sensors on the microbit. The microbit becomes a Blackbox that takes **input** from different sensors on the microbit, **processes** it and then **outputs** the information to the LED display.



Show students different Events that can be accessed from the Input toolbox.

- on A+B Pressed
- on Shake
- on Pin0
- on Face Up
- on Logo Down
- on tilt Left
- etc

Show students different sensors that can be accessed from the Input toolbox.



- Temperature
- Compass
- Light level
- Acceleration (x)
- Acceleration (z)
- Rotation (pitch)
- Rotation (roll)
- Magnetic force
- Running time (ms)
- Running time (micros)

Project Ideas, Design, & Plan

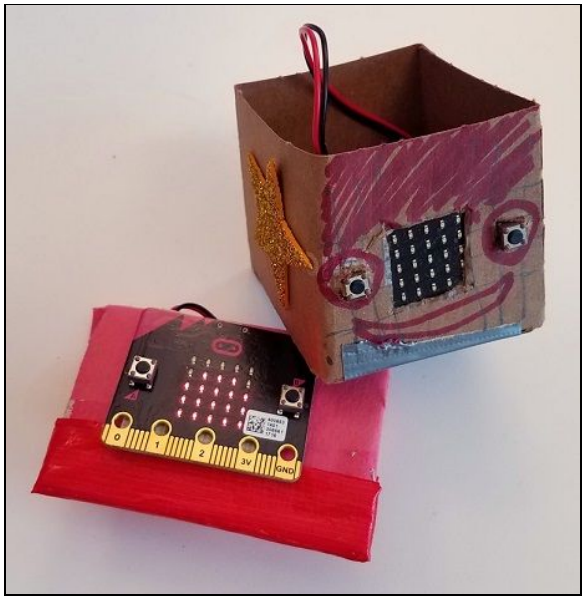
Discussion questions

- What kinds of things do they want to sense in your Blackbox? Make a list in your Notes, Plan, Reflections, & Sketches booklet.
- What inputs, processes, and outputs will be involved? Make a list in your Notes, Plan, Reflections, & Sketches booklet.
- What is your Blackbox going to look like? Sketch a design in your Notes, Plan, Reflections, & Sketches booklet.
- Create project plan using an algorithm in your Notes, Plan, Reflections, & Sketches booklet.

Remind students that a computing device has a number of inputs, and a number of outputs. The code that we write processes input by telling the micro:bit what to do when various events occur.

Project Reflection

Make a Blackbox out of the micro:bit, create an output using different sensors for each of the following events to get inputs from the sensors.



- on button A pressed
- on button B pressed
- on button A+B pressed
- on shake
- on face up, etc.

See if you can combine a maker element similar to what you created in Lesson 1 by providing a holder for the micro:bit that holds it securely when you press one of the buttons.

Sample Blackbox designs photos.

Handout for writing your reflection.
https://goo.gl/rNXMeA or “Coding & Innovation using Microbits” booklet
http://bit.ly/codingmicrobitsbooklet

Project Modifications

- Add more inputs and more outputs - use more than 4 different types of input. Try to use other types of output (other than LEDs) such as sound!
- Can the data input from the sensors be modified (processed) in a way to change the output? (Change the Celsius degrees to Fahrenheit degrees. Change different positions of the microbit to different arrows.)

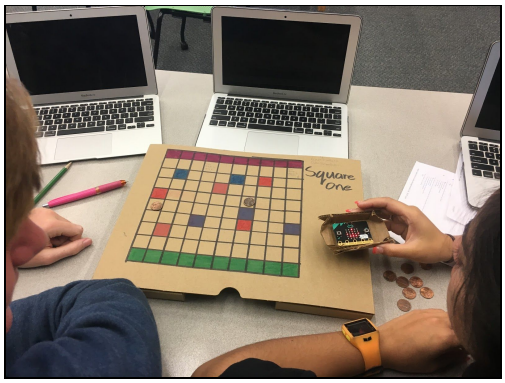
Assessment

Competency scores

Competencies	4	3	2	1
Inputs	At least 4 different inputs are successfully implemented.	At least 3 different inputs are successfully implemented.	At least 2 different inputs are successfully implemented.	Fewer than 2 different inputs are successfully implemented.
Outputs	At least 4 different outputs are successfully implemented.	At least 3 different outputs are successfully implemented.	At least 2 different outputs are successfully implemented.	Fewer than 2 different outputs are successfully implemented.
Micro:bit Program	micro:bit program: 1) uses event handlers in a way that is integral to the program 2) compiles and runs as intended 3) includes meaningful comments	micro:bit program lacks 1 of the required elements.	micro:bit program lacks 2 of the required elements.	micro:bit program lacks all of the required elements.
Collaboration Reflection	Reflection piece includes: 1) brainstorming ideas 2) construction 3) programming 4) beta testing	Reflection piece lacks 1 of the required elements.	Reflection piece lacks 2 of the required elements.	Reflection piece lacks 3 of the required elements.

03 Everything Counts (Variables)

This lesson introduces the use of variables to store data or the results of mathematical operations. Students will practice giving variables unique and meaningful names. We will also introduce the basic mathematical operations for adding, subtracting, multiplying, and dividing variables.



Lesson objectives

Students will…

- Understand what variables are and why and when to use them in a program.
- Learn how to create a variable, set the variable to an initial value, and change the value of the variable within a micro:bit program.
- Learn how to create meaningful and understandable variable names.
- Understand that a variable holds one value at a time.
- Understand that when you update or change the value held by a variable, the new value replaces the previous value.
- Learn how to use the basic mathematical blocks for adding, subtracting, multiplying, and dividing variables.
- Apply the above knowledge and skills to create a unique program that uses variables as an integral part of the program.

Lesson plan

1. **Overview**: Variables in Daily Life
2. **Unplugged**: Rock Paper Scissors
3. **Activity**: Make a Game Scorekeeper
4. **Project**: Everything Counts

Standards — CSTA K-12 Computer Science Standards

- CL.L2-03 Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities
- CT.L1:6-01 Understand and use the basic steps in algorithmic problem-solving
- CT.L1:6-02 Develop a simple understanding of an algorithm using computer-free exercises
- CPP.L1:6-05 Construct a program as a set of step-by-step instructions to be acted out
- 2-A-5-7 Create variables that represent different types of data and manipulate their values.

03.0 Introduction

Computer programs process information. Some of the information that is input, stored, and used in a computer program has a value that is **constant**, meaning it does not change throughout the course of the program. An example of a **constant** in math is '**PI**' because 'PI' has one value that never changes. Other pieces of information have values that **vary** or change during the running of a program. Programmers create **variables** to hold the value of information that may change. In a game program, a variable may be created to hold the player's current score, since that value would change (hopefully!) during the course of the game.

Ask the students to think of some pieces of information in their daily life that are **constants** and others that are **variables**.

- What pieces of information have values that don't change during the course of a single day (constants)?
- What pieces of information have values that do change during the course of a single day (variables) Constants and variables can be numbers and/or text.

Real World constants & variables

In one school day…

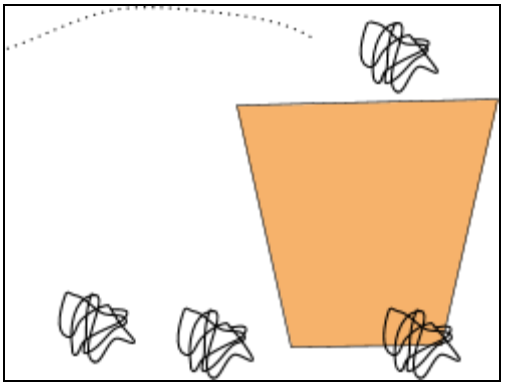
- **Constants**: The day of the week, the year, student's name, the school's address, PI in the math world
- **Variables:** The temperature/weather, the current time, the current class, whether they are standing or sitting…

Variables hold a specific type of information. The micro:bit's variables can keep track of numbers, strings, booleans, and sprites. The first time you use a variable, its type is assigned to match whatever it is holding. From that point forward, you can only change the value of that variable to another value of that same type.

- A **number variable** could hold **numerical data** such as the year, the temperature, or the degree of acceleration.
- A **string variable** holds a string of **alphanumeric characters** such as a person's name, a password, or the day of the week.
- A **boolean variable** has only two values: **true or false**. You might have certain things that happen only when the variable called *gameOver* is false, for example.
- A **sprite** is a special variable that represents **a single dot on the screen** and holds two separate values for the row and column the dot is currently in.

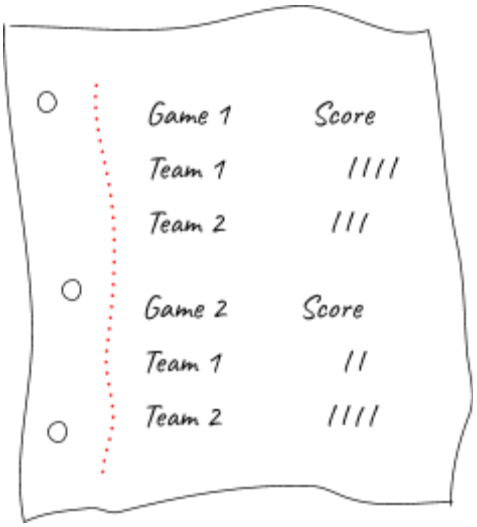
03.1 Unplugged: Keeping Score with Newspaper Toss

The objective of this activity is to experience creating and working with variables by playing crumbled *Newspaper Toss* with 2 teams in the class. *Newspaper Toss* can be played by having each student crumble a couple of newspaper pages to toss into a wastebasket or box. 1 and 2 point lines can be setup for the play. The class can be divided into 2 teams to play. Play the game a couple of times with the students keeping track of the scores. (If the class is large enough it could be divided into more then 1 game.)



Ask students to keep track of their scores on paper or in their booklet.

Sample score-keeping sheet



Ask the students what parts of the score sheet represent **constants**, values that do not change through the course of a gaming session.

Example: The team's names are constants.

Ask the students what parts of the score sheet represent **variables**, values that do change through the course of a gaming session.

Example: The teams' score of "baskets" are variables.

03.2 Activity: Counters

In a Costco store there is a person who checks to make sure anyone who enters is a member of Costco by having them show their membership card. The person checking membership cards at the door is also counting the number of people who enter the store. Costco counts the people entering the store so they have an idea of how many cashiers they will need to check people out the counter. This micro:bit activity guides the students to create a program with a variable that will keep a count of people who enter a store or an event.

Activity: People Counter

Tell the students that they will be creating a program that will act as a counter when they press a button as if they were the person counting at Costco or at a ball game. This will be able to expand to a scorekeeper by adding a counter to a second button. Students will need to create variables for the People Counter and then the Scorekeeper.

What are the variables that will be needed?

- The number of people who enter
- The number of points for team 1
- The number of points for team 2

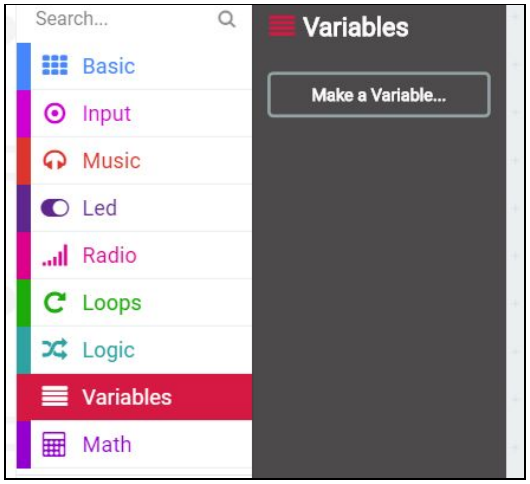
Rules for naming variables and identifiers:

- Use descriptive names
- Start with lowercase letters
- Only use letters (a-z) and numbers (0-9). No spaces or symbols
- Use camelCase when putting 2 words together
- Constants are done in all CAPS
- Math operators: +, -, *, /

Creating and naming variables:

Lead the students to create meaningful names for their variables.

- What would be a unique and clear name for the variable that will keep track of the number of people who enter?
- Student suggestions may be: ‘people’, ‘count’, ‘number’, ‘aButtonCount’…
- Discuss why (or why not) different suggestions make clear what value the variable will hold. In general, variable names should clearly describe what type of information they hold.
- Variables should start with a lowercase letter, use a capital letter if a second word is part of the variable name (camel case).
- No spaces or symbols or all CAPS.



In MakeCode, from the **Variables** menu, make and name the variable: **count**.

Initializing the variable value

It is important to give your variables an initial value. The initial value is the value the variable will hold each time the program starts. For our counter program, we will give the variable the value 0 (zero) at the start of the program.

Updating the variable value

In our program, we want to keep track of the number of people who enter. We can use the button A to do this.

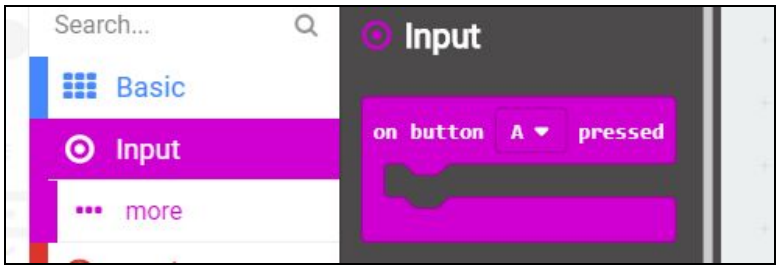
Algorithm & Pseudocode:

- *Add a comment to the onStart describing the program, coder, and date*
- *Set the count to 0 when the microbit is started*
- *Press button A to record a person entering*
- *Add another number to the variable count*
- *Display the number of people counted*

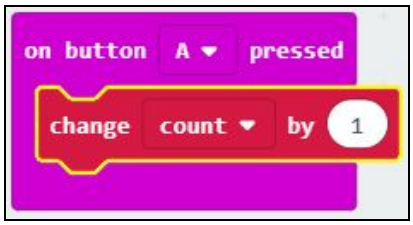
Coding People Counter

We have already created the variable **count** as described above. In the onStart the variable has been initialized to 0.

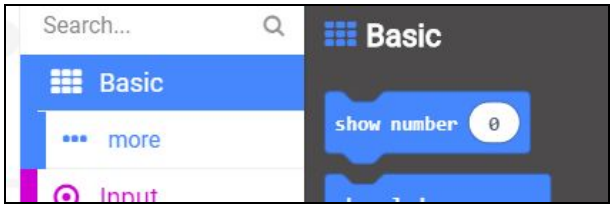
An onButtonA pressed event needs to be added to the workspace. This can be done from the Inputs toolbox menu.



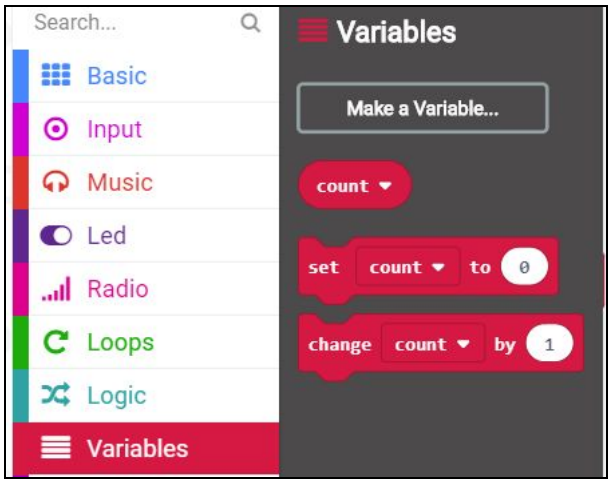
In the onButtonA press block, a block of code needs to be inserted that will increase the count by 1. This can be done by using the **change count by 1** block.

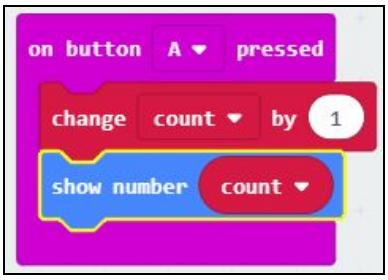


Following that block the value of **count** needs to be shown on the LEDs on the microbit. Using the Basic toolbox the **show number** block can be added below the first block of code.



In place of the “0” the variable **count** from the Variable toolbox can be place in the show number block in placed of the “0”.





Test in Simulator

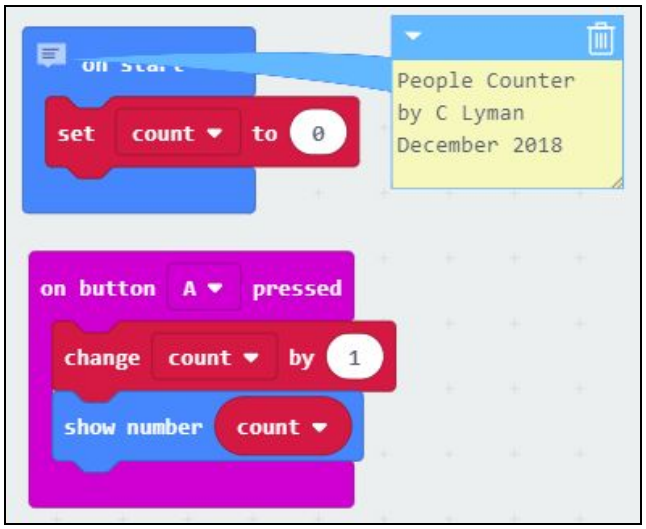
Make sure you have named the program. “People Counter” would be a good name for the program. Test the program in the **simulator** by clicking the “A” button. It should increase by 1 each time the “A” button is pressed.

Download to Microbit and test

Make any adjustments if needed. Download and test the “People Counter” on a microbit

People Counter

MakeCode blockcode:



JavaScript:

```
// People Counter
// by C Lyman
// December 2018
let count = 0
input.onButtonPressed(Button.A, function () {
    count += 1
    basic.showNumber(count)
})
count = 0
```

Sample People Counter code: https://makecode.microbit.org/_2VJCtJAMHYxM

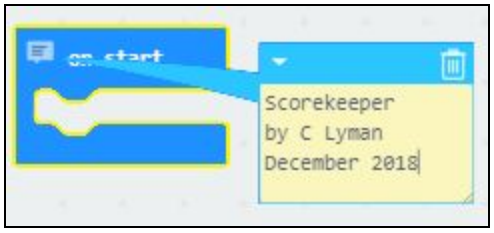
Activity: Scorekeeper

Using the concepts used in making a People Counter on a microbit, a microbit can be changed into a scorekeeper for 2 teams. The “A” button can be used to add a point for the score for Team 1 and the “B” button can be used to add a point to the score for Team 2. (Teams could also be called: Team A & Team B or Home and Guest) The current score can be displayed on the microbit after the button is pressed.

Algorithm & Pseudocode

- *Add comments to the onStart describing the program, coder, and date*
- *Create variable for each team. Such as: score1 and score2*

- Set the value for each variable to 0 when the microbit is started
- Team 1 Scores
    - Press button A to record a point for Team 1
    - Add another number to the variable score1
    - Display the score for Team 1 on the microbit
- Team 2 Scores
    - Press button B to record a point for Team 2
    - Add another number to the variable score2
    - Display the score for Team 2 on the microbit
- Display the name and score for each team
    - The A+B buttons can be setup to display the team's name and their current score
    - Display "Team 1"
    - Display the Team 1 score using the value stored in the score1 variable
    - Pause for about 1 second
    - Display "Team 2"
    - Display the Team 2 score using the value stored in the score2 variable
- Starting a new game by shaking the microbit
    - Add an onShake event
    - Clear the values of each variable by setting the values to 0
    - Display the values of each score being set to 0



Commenting code

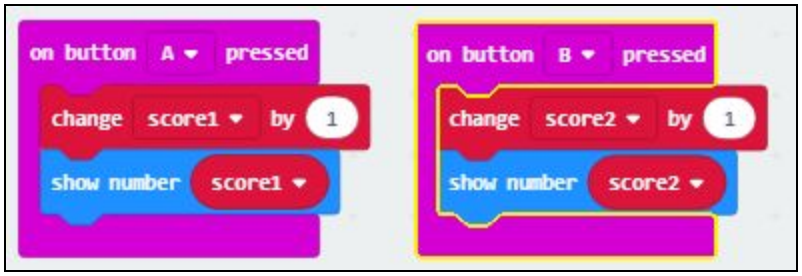
Add comments to the onStart by right clicking on the block of code. Include a title, programmer, and date in the comments.

Creating & initializing variables

Create variable for each team: score1 and score2 and then set the value for each variable to 0 when the microbit is started.

Adding points for each team

Set up an onButton A pressed event to record a point for Team 1. Have it add another number to the variable **score1** when the A button



is pressed. Then display the score for Team 1 on the microbit.

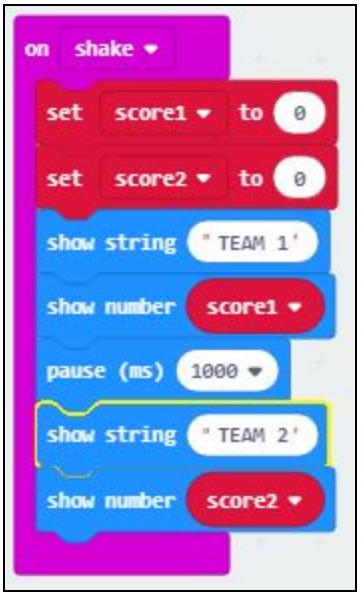
Set up an onButton B pressed event to record a point for Team 2. Have it add another number to the variable **score2** when the B button is pressed. Then display the score for Team 2 on the microbit.

Displaying scores for both teams

Display the name and score for each team. Use the onButton A+B event to display the each teams name and the current score. Use a showString block for “Team 1” and then a showNumber of the value stored in score1 variable. Use a pause block for 1 second (1000 ms). Then use a showString block for “Team 2” and then a showNumber of the value stored in score2 variable.

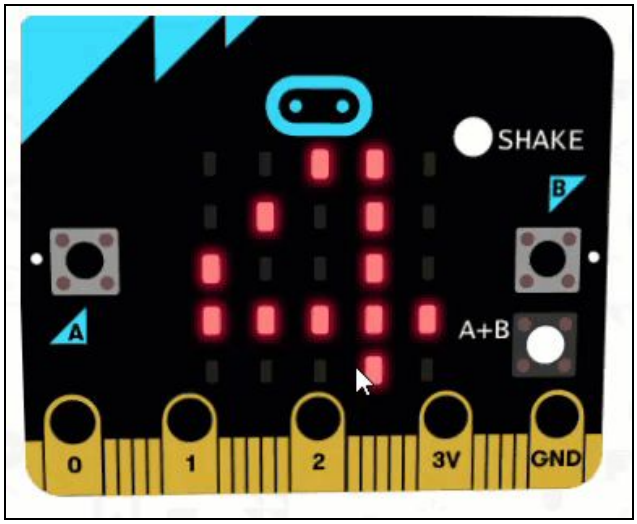
Clear the scores

Just as an “Etch-a-Sketch” is cleared by turning it upside down and shaking it, have the microbit clear the current scores and set them back to 0 by shaking the microbit. Add an onShake event and clear the scores by setting score1 and score2 to 0 and then displaying the 0 score to the microbit. At this point the microbit is ready to keep score for a new game.



Test it out!

In the simulator test the project by clicking the “A” and the “B” buttons to add point to each teams score. Test the “A&B” to display the current scores and name for each team. Shake the simulator or click the “Shake” button to reset the scores and start a new game.



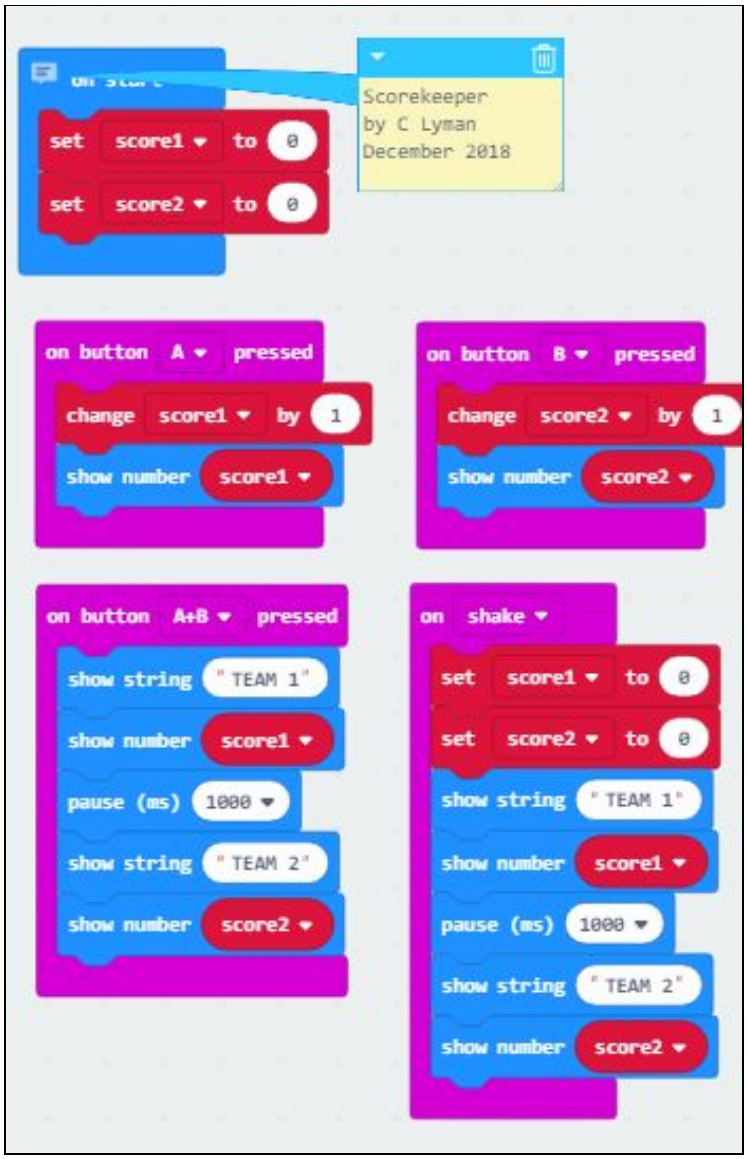
Link to Youtube video of the simulator:
https://youtu.be/H732aNH9EVQ

Download to microbit and test

Link to sample project:
https://makecode.microbit.org/_C7We3bW9ohd3

Scorekeeper Code

MakeCode block version



JavaScript version

```
let score2 = 0
let score1 = 0
input.onButtonPressed(Button.A, function () {
    score1 += 1
    basic.showNumber(score1)
})
input.onButtonPressed(Button.B, function () {
    score2 += 1
    basic.showNumber(score2)
})
input.onButtonPressed(Button.AB, function () {
```

```
    basic.showString("TEAM 1")
    basic.showNumber(score1)
    basic.pause(1000)
    basic.showString("TEAM 2")
    basic.showNumber(score2)
})
input.onGesture(Gesture.Shake, function () {
    score1 = 0
    score2 = 0
    basic.showString("TEAM 1")
    basic.showNumber(score1)
    basic.pause(1000)
    basic.showString("TEAM 2")
    basic.showNumber(score2)
})
score1 = 0
score2 = 0
```

Play “Newspaper Toss” and keep score on the microbit

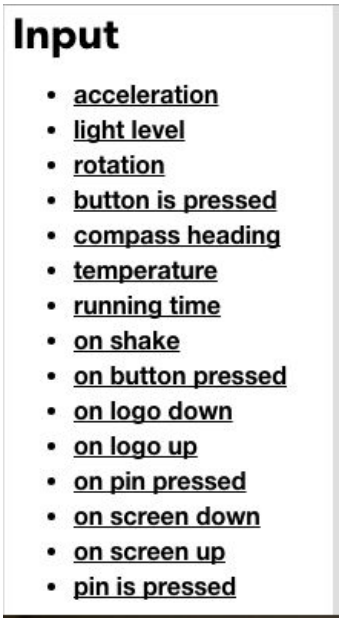
Divide the class up into team and have them each keep score on their microbit as they play “Newspaper Toss”. When the game is over each student should have the same scores on their microbit.

Modifications

- Change team names to “Home” and “Guests”
- Build a box for the scorekeeper
- Add a game type title in the onStart

03.3 Innovation Project: Everything Counts

This is an assignment for students to come up with a micro:bit program that counts something. Their program should keep track of **input** by storing values in variables, and provide **output** in some visual and useful way. Students should also perform mathematical operations on the variables to give useful output.



Inputs

Remind the students of all the different inputs available to them through the micro:bit.

Project ideas:

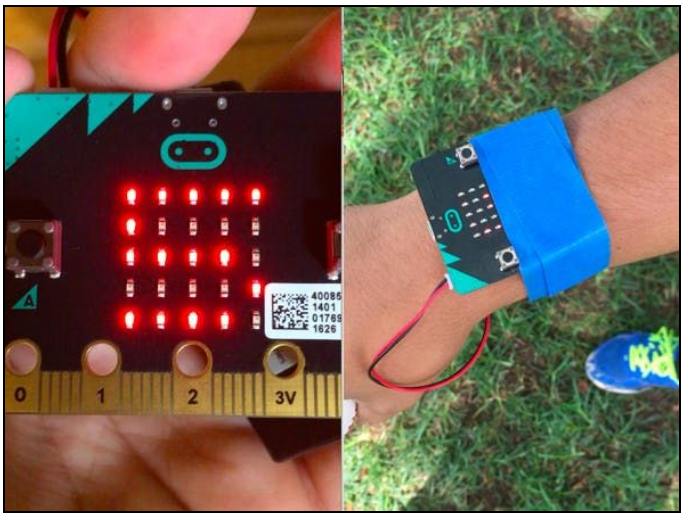
Duct tape wallet

You can see the instructions for creating a durable, fashionable wallet or purse out of duct tape: Duct tape wallet. (https://makecode.microbit.org/projects/wallet) Create a place for the micro:bit to fit securely. Use Button A to add dollars to the wallet, and Button B to subtract dollars from the wallet.

Extra modification: Use other inputs to handle cents, and provide a way to display how much money is in the wallet in dollars and cents.

Umpire's baseball counter (pitches and strikes)

In baseball during an at-bat, umpires must keep track of how many pitches have been thrown to each batter. Use Button A to record the number of balls (up to 4) and the number of strikes (up to 3).



Extra modification: Create a way to reset both variables to zero, create a way to see the number of balls and strikes on the screen at the same time.

Population Survey Counter

A lot of times 2 different traits or kinds objects need to be counted for a comparison ratio. (male vs female, blond vs brown/black hair, holey jeans vs no holes, trucks vs cars, white vs other colored cars, etc.) Different traits in a population can be tallied by

marking lines on a piece of paper or a microbit can be programmed to keep the tallie by pressing buttons.

Shake counter

Using the ‘On Shake’ block, you can detect when the micro:bit has been shaken and increment a variable accordingly. Try attaching the micro:bit to a lacrosse stick and keep track of how many times you have successfully thrown the ball up in the air and caught it.

Extra modification: Make the micro:bit create a sound of increasing pitch every time you successfully catch the ball.

Pedometer

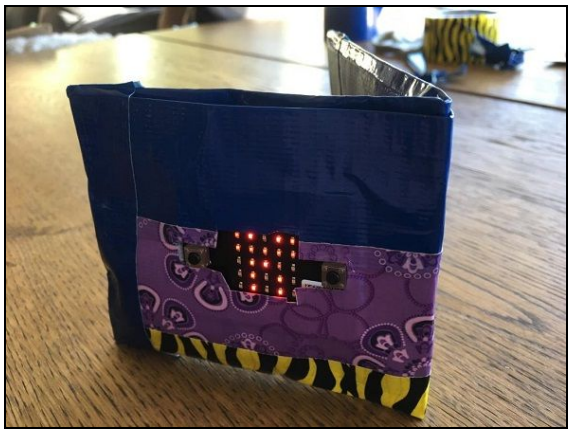
See if you can count your steps while running or doing other physical activities carrying the micro:bit. Where is it best mounted? Try using a gravity setting instead of just onShake.

Extra Modification: Design a wearable band or holder that can carry the micro:bit securely so it doesn’t slip out during exercise.

Calculator

Create an adding machine. Use Button A to increment the first number, and Button B to increment the second number. Then, use Shake or Buttons A + B to add the two numbers and display their sum.

Extra modification: Find a way to select and perform other math operations.



Duct tape wallet with micro:bit display

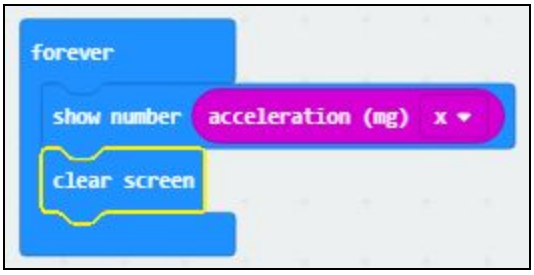
Project Ideas, Design, & Plan

In any design project, it's important to start by understanding the problem. You can begin this activity by interviewing people around you who might have encountered the problem you are trying to solve. For example, if you are designing a wallet, ask your friends how they store their money, credit cards, identification, etc. What are some challenges with their current system? What do they like about it? What else do they use their wallets for?

If you are designing something else, think about how you might find out more information about your problem through interviewing or observing people using current solutions.

Then start brainstorming. Sketch out a variety of different ideas. Remember that it's okay if the ideas seem far-out or impractical. Some of the best products come out of seemingly crazy ideas that can ultimately be worked into the design of something useful. What kind of holder can you design to hold the micro:bit securely? How will it be used in the real world, as part of a physical design?

Use the simulator to do your programming, and test out a number of different ideas. What is the easiest way to keep track of data? If you are designing for the accelerometer, try to see what different values are generated through different actions (you can display the value the accelerometer is currently reading using the 'Show Number' block; clear the screen afterward so you can see the reading).



Reflection

Have students write a reflection of about 150–300 words, addressing the following points:

- What was the problem you were trying to solve with this project?
- What were the Variables that you used to keep track of information?
- What mathematical operations did you perform on your variables? What information did you provide?
- Describe what the physical component of your micro:bit project was (e.g., an armband, a wallet, a holder, etc.)
- How well did your prototype work? What were you happy with? What would you change?
- What was something that was surprising to you about the process of creating this project?
- Describe a difficult point in the process of designing this project, and explain how you resolved it.

Reflection can be written in the student "Coding & Innovation using Microbits" booklet or the handout for help writing your reflection. https://goo.gl/Yj1dzw or "Coding & Innovation using Microbits" booklet http://bit.ly/codingmicrobitsbooklet

Assessment

Competency scores

Competency	4	3	2	1
Variables	At least 3 different variables are implemented in a meaningful way.	At least 2 variables are implemented in a meaningful way.	At least 1 variable is implemented in a meaningful way.	No variables are implemented.
Variable Names	All variable names are unique and clearly describe what information values the variables hold using CamelCase	The majority of variable names are unique and clearly describe what information values the variables hold.	A minority of variable names are unique and clearly describe what information values the variables hold.	None of the variable names clearly describe what information values the variables hold.
Mathematical Operations	Uses a mathematical operation on at least two variables in a way that is integral to the program.	Uses a mathematical operation on at least one variable in a way that is integral to the program.	Uses a mathematical operation incorrectly or not in a way that is integral to the program.	No mathematical operations are used.
Micro:bit Program	micro:bit program: 1) Uses variables in a way that is integral to the program 2) Uses mathematical operations to add, subtract, multiply, and/or divide variables 3) Compiles and runs as intended 4)Meaningful comments in code.	micro:bit program lacks 1 of the required elements.	micro:bit program lacks 2 of the required element.s	micro:bit program lacks 3 or more of the required elements.
Collaboration Reflection	Reflection piece addresses all prompts.	Reflection piece lacks 1 of the required elements.	Reflection piece lacks 2 of the required elements.	Reflection piece lacks 3 of the required elements.

04 Making Decisions (Conditional)

This lesson introduces the Logic blocks such as 'If…then' and 'If…then…else'. Students practice skills of creativity, problem-solving, and collaboration.

Lesson objectives

Students will…

- Understand what conditional statements are, and why and when to use them in a program.
- Learn how to use the Logic blocks 'If…then' and 'If…then…else'.
- Practice using the Logic blocks so different conditions yield specified outcomes.
- Demonstrate understanding and apply skill by collaborating with classmates to create a game that uses a micro:bit and a program that correctly and effectively uses conditionals.

Lesson plan

1. **Overview**: Conditional statements
2. **Unplugged**: Red light, green light
3. **Activity**: Rock, paper, scissors
4. **Project**: Board game

Standards — CSTA K-12 Computer Science

- CL.L2-03 Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities.
- CL.L2-04 Exhibit dispositions necessary for collaboration: providing useful feedback, integrating feedback, understanding and accepting multiple perspectives, socialization.
- CL.L3A-01 Work in a team to design and develop a software artifact.
- K-12 Computer Science Framework Core concept: Control Structures

04.0 Introduction

Computer programs are instructions telling the computer how to process input and deliver output. An important part of programming is telling the computer WHEN to perform a certain task. For this, we use something called 'conditionals'. Conditionals get their name because a certain **Condition** or Rule has to be met.

Students are all already familiar with the concept of conditionals in their daily lives!

Have they ever had their parents say..?

- “If you clean your room, you can go out with your friends.”
- “If your homework is done, you can play video games.”
- “If you don’t wear green on St. Patrick’s Day you could get pinched!”
- “If you do your chores all week, you get your allowance, else you are grounded.”

These are all conditionals! Conditionals follow the format of IF this, THEN that.

If … then structures

IF (condition is met) **THEN**
(action performed)

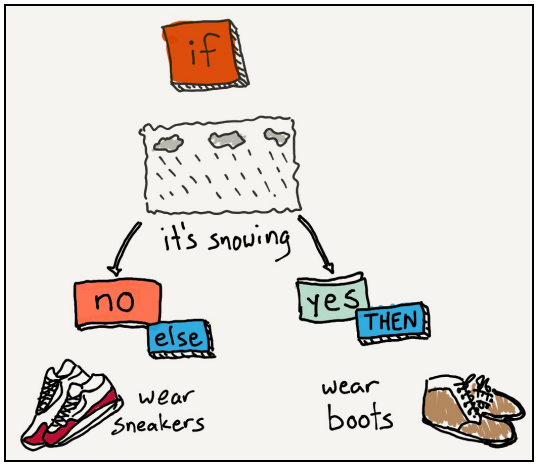
Have the students share a few conditionals from their own lives with the class or within small groups.

Note: For older students, you can have them add the ELSE portion of a conditional.

IF (condition is met) **THEN**

(action performed),
ELSE

(different action performed)



Example:

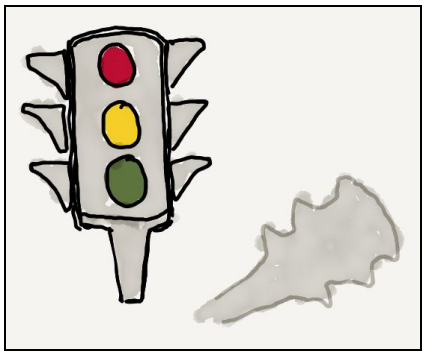
- IF (it is snowing) THEN
  wear boots
  ELSE
  wear shoes.

The ELSE portion makes sure that a different action is performed in either case. Without the ELSE action, your students might be barefoot!

Tell the students that they will be acting out some conditionals as though the whole class is a computer program for a game. Each student will perform a described action if the indicated condition is met.

Note: This activity can be done as a whole class or in smaller groups or as a pencil and paper activity.

04.1 Unplugged: Red Light, Green Light



Objective

To reinforce the programming of basic conditionals by having students experience conditionals through acting them out in real life.

Activity overview

Students will line up at one end of the classroom with the goal of reaching the other side of the classroom. The teacher, and then the students themselves will call out conditionals and all the students will advance or not depending on the specific conditional statement.

Note: As the teacher you will need to keep an eye out for any 'errors' that occur during the running of the program.

Materials

- Pencils and lined paper (if doing this activity seated). Students can advance across the paper instead of the room with one inch line equal to one step.

Process

- Have the students line up at one side of the room.
- Explain the rules:
    - The object of the game is to get across the room first.
    - For if...then conditionals: If the condition called out is true for you, then perform the action described in the then. If the condition called out is false for you, then do nothing.
    - For if...then...else conditionals, listen carefully to the whole condition, as the else may apply to you.

Example conditional statements

- If you are wearing something green, then take a step forward.
- If you have the letter 'e' in your first name, then take two giant steps forward.
- If you are wearing sneakers, then take a step forward, else take 2 steps forward.
- If your birthday is this month, then take a giant hop forward. The conditionals you use will depend on your individual class.

After the students get the idea of the game, allow them to make up and call out conditionals (that meet teacher approval).

They will need to be observant, as a conditional that moves them forward, will also move their competition forward!

Tips

- SAFETY FIRST! Students, especially younger ones, can get quite silly with this and while it is meant to be fun and even funny, safety first!
- Student conditionals need to apply to at least two people in the class.

Reflections

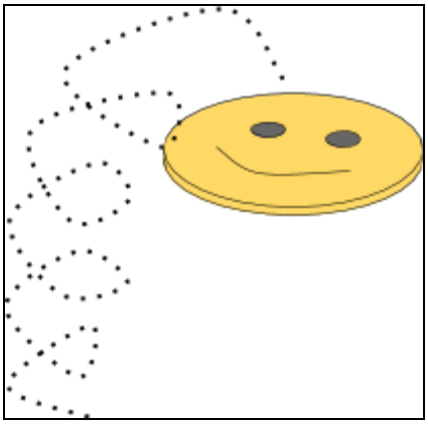
How did they do? Were there any 'run-time errors'? Did a student miss a conditional being met or fail to correctly carry out the THEN or ELSE action? Were there some conditions that could be evaluated as something other than True or False (maybe, sometimes)? List 3 of the different commands using while playing "Red Light, Green Light". This can be done on a piece a paper on in the Reflection Booklet.

Extensions/Variations

- Add AND, OR, AND/OR statements to the conditionals.
- Example: If you have brown hair AND brown eyes, then…
- Create nested IF's
- Example: If you are wearing sneakers, then… if you are also wearing white socks, take three steps forward.
- Let students create their own conditionals for future program runs with the class. (A very popular activity, though all conditionals should be run by the teacher first for approval.)
- Relate this activity to a system and have the students create the conditionals that would end in a product of some kind or the completion of some task, like writing a sentence or setting a table or constructing a simple structure.

04.2 Activity: Coin Toss

For this activity, each student will need a micro:bit. Everyone will create the same programs, the classic "Heads or Tails" coin toss and a "Dice Roll".



Introduce activity

- Have students recall the start of a ball game to see who gets to start with the ball. The referee asks the players to call "Heads or Tails".
- What are the rules of the game? What are the conditionals?
- Example: If Player A calls "Heads" and the toss is "Heads" they get the ball, Else Player B gets the ball or wins the toss..

- Have students write the pseudocode for how to determine the winner of a coin toss on the micro:bit.

Algorithm & Pseudocode:

- *On shake:*
  - *Choose random number from 0-1*
  - *If random number = 0, then display Heads icon,*
  - *Else if random number = 1, then display Tails icon.*
- *Point out that because there are only 2 possibilities, we don't need to do a separate check to see if random number = 1. So we just use an else.*

Coding Coin Toss on the micro:bit

In the MakeCode programming environment start a new project. Name the project "Coin Toss". Pair up the microbit with the browser if the firmware has been updated and you are using a Chrome browser. (Click on the "gear" and pair device.) The forever loop event can be drug back to the toolbox as it will not be needed.

Commenting code

In the onStart event add a comment naming the program, the coder, and the date of the project.

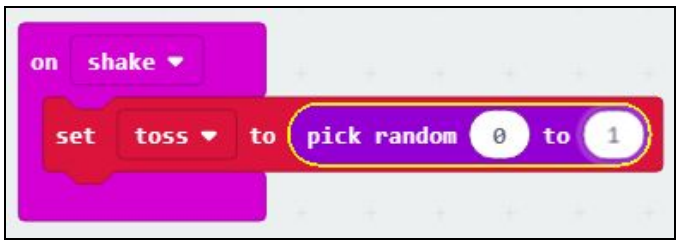
Creating & initializing the variable

A variable can be created in the Variables toolbox. It can be called ***toss*** and given a starting value of **0** in the onStart event.

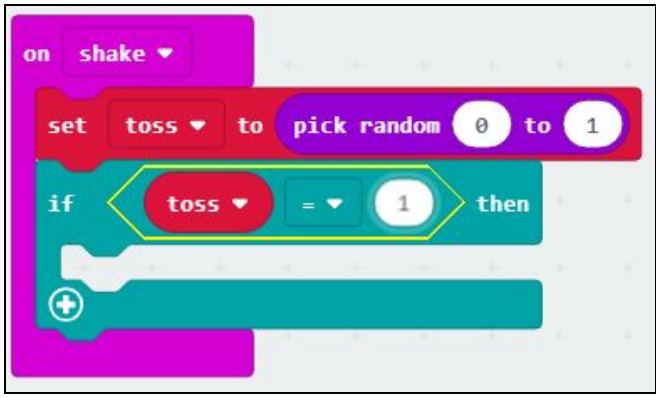
Decide to display "heads" or "tails"

From the Input toolbox add an **onShake** event to the workspace. From the Variables toolbox add a **"set toss to 0"** block. From the Math toolbox replace the "0" with a **"pick random 0 to 10"** block and change the "10" to a "1". This will generate a random number either 0 or 1 and store it in the

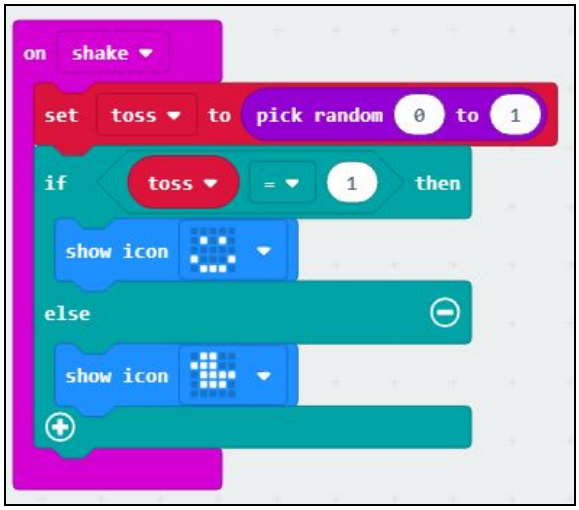
variable ***toss***.



Next we will use a "**if … then … "** statement to make a decision whether to show "heads" or "tails" on the microbit. From the Logic toolbox add an **"if (true) then**" block. Under the set "toss" … statement. From **Logic toolbox** add a comparison **"0 == 0"** block in place of the "true" comparison. (In most programming languages the **"=" is used as the assignment operato**r and the **"==" is used to compare two items to see if they are equal.**) Change the first 0 to the variable "toss" and the second 0 to the number 1. The statement now reads, "if (toss == 1) then …" do something. In coding the "0" is usually associated with "false" and the "1" is associated with "true".

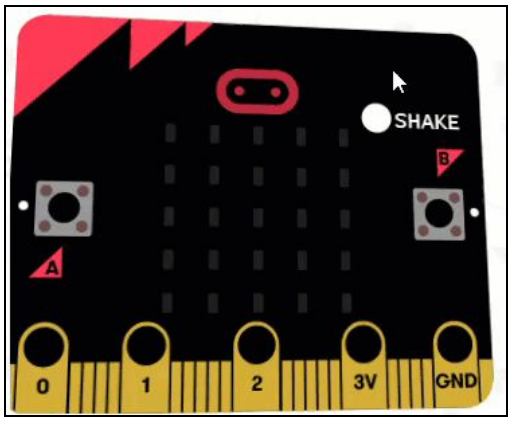


Add a "**showIcon"** block and choose a "smiley" face. At the bottom-left of the "if (condition) then ... " there is a "**+**", by clicking on the "+" the block is changed to an **"if (condition) then … else…**" block. In the "**else**" section of the block drag a "**show icon**" block and choose an icon for the tail. Now when the microbit is shaken, a number is randomly generated, 0 or 1. If it is a 1 it will show a smiley face, representing "heads" else when a 0 is chosen an image representing "tails" is displayed.



Test it out!

Test it out in the simulator by shaking it with your mouse or clicking on the SHAKE button. You should see both the “heads” and “tails” icons show up with enough tries. Make any adjustments to your code needed.



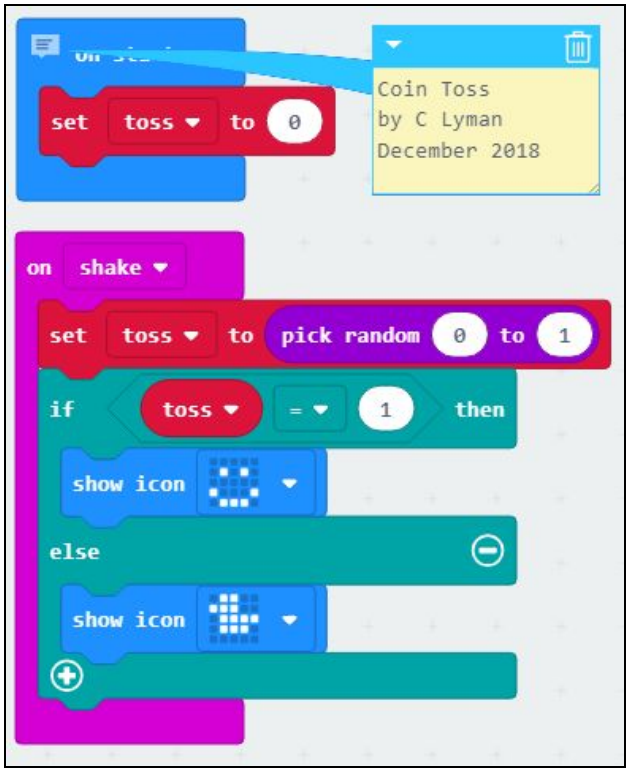
if

Download to microbit and test

Once you have it working to your satisfaction, download it to the microbit and test it there. Keep a tally to see if shakes about even “heads” and “tails”.

Coin Toss code

MakeCode block code



JavaScript

```
// Coin Toss
// by C Lyman
// December 2018
let toss = 0
input.onGesture(Gesture.Shake, function () {
    toss = Math.randomRange(0, 1)
    if (toss == 1) {
        basic.showIcon(IconNames.Happy)
    } else {
```

```
            basic.showIcon(IconNames.Duck)
        }
    })
    toss = 0
```

Sample **Coin Toss** program: https://makecode.microbit.org/_3U99agcwr4UL

Play "Coin Toss" and keep a tally of the "heads" and "tails"

Keep a tally to see if shakes about even "heads" and "tails".

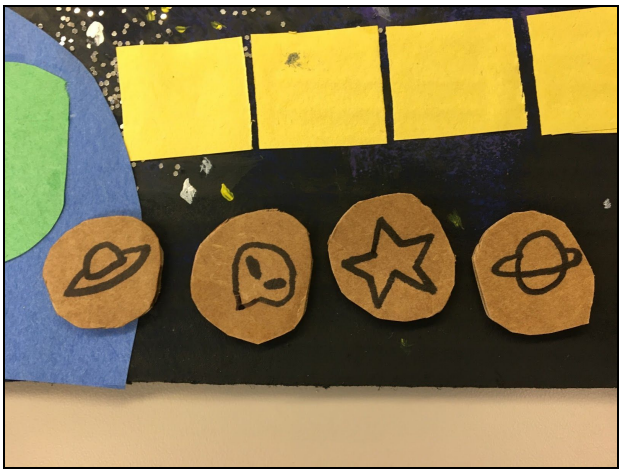
- Working from the specifications, have students work to try to code a "Heads Tails" game on their own.

Ideas for modifications

- Have the program clear the display after a second or so.
- Modification: Have the microbit keep track of "heads" and "tails"
- Similar to what was done in the Scorekeeper program in the previous module.
- Change the microbit so it works as a **dice roll** when it is shaken and displays the 6 different dice faces when an associated random number is chosen. (Hint: get a random number between 0-5 and add 1 to it. Use a different 'if (condition) action' statement for each dice face.)

04.3 Innovation Project: Board Game

This is an assignment for students to create a board game. It should take two to three class periods. If your school has a makerspace or an art classroom where students can access materials such as cardboard, poster paints, or markers, you might schedule your classes to work there. Have the students plan their project in their student booklet in the space provided for this project.



Once students have finished the first version of their games, schedule time for students to play each other's games. Ideally, give them some time to give and gather feedback, then revise their games accordingly.

Introduction

Many board games use an electronic toy to signal moves, or provide clues. There are some funny examples online if you search for "electronic board game". Here are some examples:

Dark Tower (featuring Orson Welles): This is an example of a circular board game in which the pieces start on the edges and move in toward the middle. https://youtu.be/cxrY7MWEkwE

Electronic Dream Phone Board Game Commercial - 1992: This board game is really a logic puzzle. There are printed clues that illustrate relationships and the phone provides clues that help you to narrow down possibilities by a process of elimination.
https://www.youtube.com/watch?v=pqYsQgDqlmg

Stop Thief Electronic Board Game commercial 1979: This board game uses a device to give audio clues that help you to figure out what to do on the game board. It's a good example of how you might use sound as a clue. https://www.youtube.com/watch?v=q3wpPRdDy4E

Project Ideas, Design, & Plan

Assignment

Students should work in pairs to create an original board game project in which micro:bit is a central feature, and the rules of their board game should use Conditionals.

Students will need to work together to come up with:

- A set of written rules (how to play)
- A game board
- A program for the micro:bit
- Photo documentation of the different game pieces, cards, or other components of the game with the micro:bit included as well as a screenshot of your micro:bit code. Each photo must have a caption that describes what the photo is documenting.
- Reflection: A text entry describing your team's game making process and each teammate's part in the creation of the game from brainstorming ideas, through construction, programming, and beta testing.

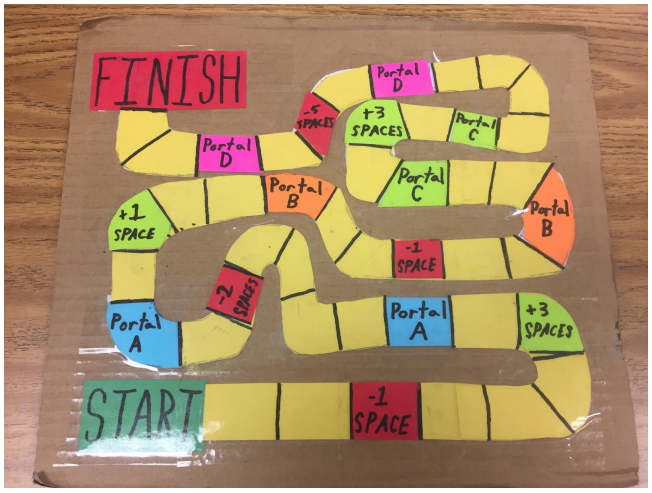
The micro:bit needs to work in conjunction with the game board and/or game pieces and should be a central feature of the game. *Ideally, it should be more than a simple substitute for a six-sided die.*

The micro:bit might:

- Simulate the results of a battle between two pieces
- Randomly point in a different direction of travel
- Generate a result based on its current incline
- Point randomly at players and stop them
- Randomly display a letter and a number for a coordinate on a gameboard
- Display a dynamic score
- … let your imaginations run wild!

Ideally, students should be writing their own versions of micro:bit programs to do something original.

Board game examples:



Teleportation game

Here a sample board game and code by Cameron D.:

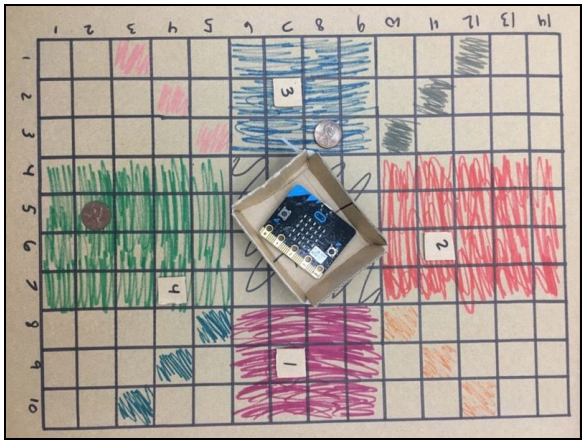
https://makecode.microbit.org/_EE4TPkdMVJtz

The microbit that is used to play the game is programmed when it is shaken to give a random number between 1-4 and then an arrow that points forward or backwards. That tells the player how far to move and in what direction. If the player lands on a portal they are transported to the other opening of the portal. This could be both forward or backwards. The object of the game is to move from the Start to the Finish.

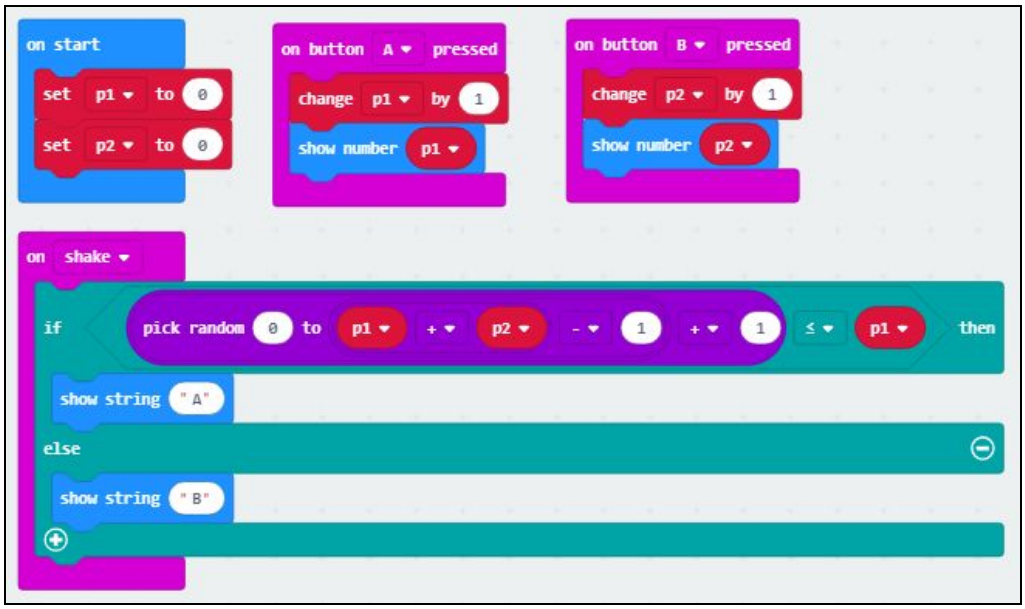
Here is one simple program to discuss and use as an example:

Battle pieces

In this example, pieces start out at full strength and lose points based on random events on the board. When two pieces meet on the same space, they battle.



- Press A to enter the strength of piece A.
- Then press B to enter the strength of piece B.
- Shake the micro:bit to determine the winner of the battle, which is proportionately random to the strength of each piece.



Link to MakeCode program: https://makecode.microbit.org/_4wyhFTigHMYr

Beta testing

Give students a chance to play each other's games. The following process works well:

- Have each pair of students set up their own project at their table.
- Leave a clipboard or a laptop on the table for taking notes.
- Rotate the students through each project, moving clockwise around the room:
    - Play the game (5 min)
    - Fill out a survey form (5 min)

Sample Survey questions

- How easy was it to figure out what to do?
- What is something about this project that works really well?
- What is something that would make this project even better?
- Any other comments or suggestions?

Many online survey tools will allow you to sort the comments by project and share them with project creators so they can make improvements based on that feedback.

Reflection

Have students write a reflection of about 150–300 words, addressing the following points:

- Explain how you decided, as a pair, on your particular board game idea.
- What was something that was surprising to you about the process of creating this game?

- Describe a difficult point in the process of designing this game, and explain how you resolved it.
- What feedback did your beta testers give you? How did that help you improve your game? What were the Conditionals that you used as part of your game rules?

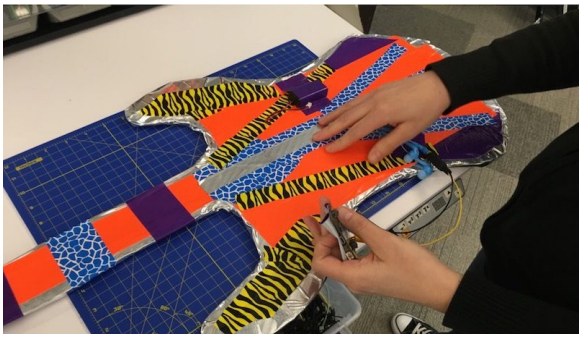
Use Student Coding & Innovation Booklet or use the handout to help writing your reflection.
https://goo.gl/PyYbNm or “Coding & Innovation using Microbits” booklet
http://bit.ly/codingmicrobitsbooklet

Assessment

Competency scores

Competency	4	3	2	1
Rules	All game rules are clear and complete.	A game rule is missing or not complete or not clear.	More than one game rule is missing or not complete or not clear.	Most of the game rules are missing or it is not clear what the rules are.
Game Board	Game board is: 1) Complete 2) Neat 3) Fits with the theme of the game 4) micro:bit is a central part of the game	Game board meets only 3 of the conditions listed for a score of 4.	Game board meets only 2 of the conditions listed for a score of 4.	Game board meets only 1 of the conditions listed for a score of 4.
Micro:bit Program	micro:bit program: 1) Uses the micro:bit in a way that is integral to the game 2) Uses conditionals correctly 3) Compiles and runs as intended 4) JavaScript includes comments in code	micro:bit program lacks 1 of the required elements.	micro:bit program lacks 2 of the required elements.	micro:bit program lacks 3 of the required elements.
Photo Documentation	Complete photo documentation that includes photos of game board and code and captions.	A photo is missing or of poor quality or a caption is missing.	Multiple photos and/or captions missing or of poor quality.	Most photos and/or captions missing or of poor quality.
Collaboration Reflection	Reflection piece includes: 1) Brainstorming ideas 2) Construction 3) Programming 4) Beta testing	Reflection piece lacks 1 of the required elements.	Reflection piece lacks 2 of the required elements.	Reflection piece lacks 3 of the required elements.

05 Music, Designs, & LEDs (Loops)



This lesson introduces the concept of looping and iteration. In the lesson the concepts of using pins for buttons, connecting headphones or speakers to output music or sound, and the ability to connect external LEDs to the microbit through the external pins are introduced. By understanding how to connect external devices to the microbit it opens up a whole new world that can lead to all kinds of ***innovation.*** *Think of self driving cars that take inputs from sensors, process the data, and output the results by steering, driving, and braking the car. Smart homes with sensors, processors, and lights coming on or the heat being controlled are different kinds of innovation that a microbit can be a starting point for* ***innovation.***

Different kinds of loops will be introduced and implemented.

Lesson objectives

Students will…

- Understand the value of looping (iteration) in programming
- Understand looping as a form of iteration
- Learn how and when to use the Looping blocks ‘repeat’, ‘while’, and ‘for’
- Apply the above knowledge and skills to create a unique program that uses iteration and looping as an integral part of the program
- Think of how a microbit could lead to new types of innovation

Lesson structure

- Introduction: Lather. Rinse. Repeat.
- Unplugged Activity: Walk a Square pseudocode
- micro:bit Activities: Code a Heart Beat program, code a 2 tone siren, code the song “Frere Jacques”, code a number counter
- Project: Loopy Entertainment and Innovation!
- Project Mods: Use LED lights or servo motors to add a maker element to the project
- Assessment: Rubric
- Standards: Listed

Lesson plan

1. **Overview**: Looping
2. **Unplugged**: Walk a square
3. **Activity**: Loops demos

4. **Project**: Loopy Entertainment and Innovation!

Standards — CSTA K-12 Computer Science Standards

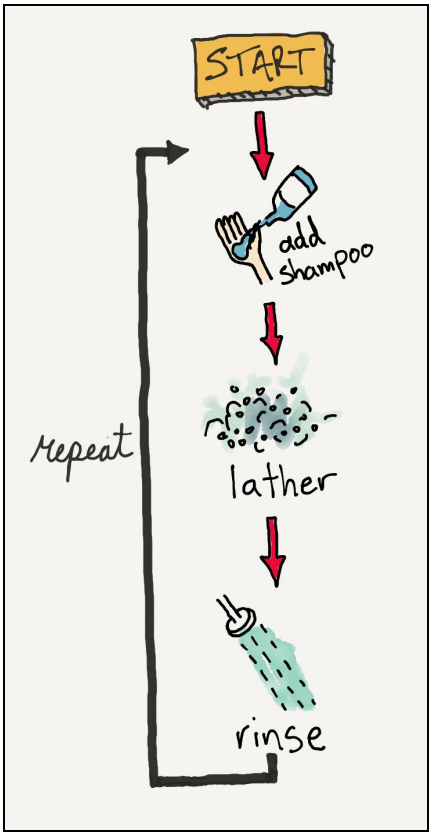
- CL.L2-05 Implement problem solutions using a programming language, including: looping behavior, conditional statements logic, expressions, variables, and functions.
- CL.L3A-03 Explain how sequence, selection, iteration, and recursion are building blocks of algorithms.

05.0 Topic Introduction

In computer programming, iteration is the repetition of a sequence of code. A loop is a form of iteration. A loop repeats code until a certain condition is met.

Questions for the students:

- Do you use shampoo to wash your hair? *Most will say 'Yes'*.
- Have you ever read the instructions on a bottle of shampoo? *Most will say 'No'*.



Most of us have never read the instructions on a bottle of shampoo, because we already know how to use shampoo.

What algorithm could you write for shampooing your hair? Example:

1. Wet hair.
2. Apply shampoo to wet hair
3. Scrub shampoo into hair
4. Rinse shampoo out of hair

If you did read the instructions on a bottle of shampoo, you may read similar instructions as the ones you just wrote with one added step at the end. That step is 'Repeat.' How does this one extra step affect the algorithm?

In computer programming, this is known as the 'shampoo algorithm' and is an example of a loop. It is also an example of an 'infinite' or 'endless' loop as the algorithm keeps repeating with no condition that ends the looping.

http://DBwebsolutions.com

'Rinse. Repeat.' has even become a meme and made its way into modern song lyrics. What other common activities involve repetitive actions? *Examples: Singing (choruses repeat), dancing, school cheers, walking and running, exercise routines…*

Optional - Lather, Rinse, Repeat

Share with your students the history of '**Lather, Rinse, Repeat**.'

Lather, Rinse, Repeat: Hygiene Tip or Marketing Ploy By Lauren Goldstein October 11, 1999 http://archive.fortune.com/magazines/fortune/fortune_archive/1999/10/11/267035/index.htm (FORTUNE Magazine) – In Benjamin Cheever's novel The Plagiarist, a marketing executive becomes an industry legend by adding one word to shampoo bottles: REPEAT. He doubles shampoo sales overnight.

This bit of fiction reflects a small yet significant eddy of U.S. consumer angst: If we REPEAT, are we or are we not playing into the hands of some marketing scheme? It turns out that in real life there's a reason you should repeat, or at least there used to be. In the 1950s, when shampoos began to be mass-marketed, we didn't wash our hair all that often–once or twice a week, as opposed to five times a week as most of us do now. Also, we used a lot more goop in our hair. It was the age of Brylcreem and antimacassars, remember. Paul Wallace, the director of hair-care research and development for Clairol, says that when cleaning agents in shampoo came up against that amount of oil and goop, "it depressed the lather." A second application was needed to get the suds that consumers expected. Lots of suds mean that hair is already clean. Maybe too clean (there's no oil to break through), but consumers like it.

FORTUNE asked Frederic Fekkai, the noted and notably expensive New York City hairdresser, what he thought about the double lather. He says, "Yesterday I put oil on my hair for a different look and went to a restaurant where the smoke was horrible. This morning I realized I had to do two shampoos."

At any rate, Wallace says advances in shampoo technology mean that only one application of, for instance, Clairol's Herbal Essences is sufficient to break through the oiliest hair. The company has stricken the use of both REPEAT and REPEAT IF DESIRED from all Clairol products. Yet a lot of brands, like Suave by Unilever and L'Oreal, still say REPEAT. Others, like Unilever's Finesse and Revlon's Flex, opt for the less imperative REPEAT IF DESIRED. Procter & Gamble uses REPEAT IF NECESSARY on Pantene.

Getting consumers to wash twice can, of course, increase sales–in ways one might not imagine. Double sudsing leads to dry hair, Fekkai points out, and that means more beauty products! "When you do two shampoos, even if you don't usually use a conditioner, you have to use a little," he says.

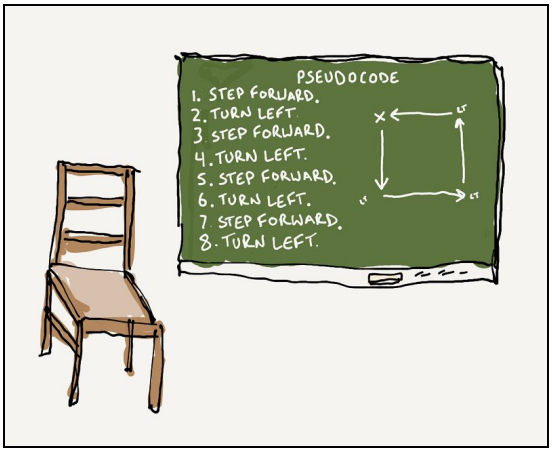
"The conditioner becomes very important." REPEAT. FOLLOW WITH CONDITIONER. Words Cheever's marketer could have retired on. –Lauren Goldstein

From Wikipedia (https://en.wikipedia.org/wiki/Lather,_rinse,_repeat): Lather, rinse, repeat (sometimes wash, rinse, repeat) is an idiom roughly quoting the instructions found on many brands of shampoo. It is also used as a humorous way of pointing out that such instructions if taken literally would result in an endless loop of repeating the same steps, at least until one runs out of shampoo. It is also a sarcastic metaphor for following instructions or procedures slavishly without critical thought.

05.1 Unplugged: Walk a square

Objective

To reinforce the concept of looping (iteration) by having students act out the repeated steps of an algorithm in real life.

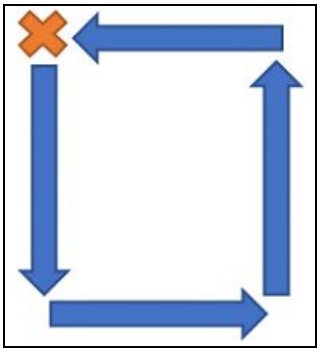


Overview

Students will give the teacher instructions to do a simple activity, then look for places where using iteration could shorten their code and make it more efficient.

Process

- Place a chair in the front of the room.
- Stand at the back right side of the chair facing the students.
- Ask the students what instructions they could give you that when followed would lead you to walk around the chair, ending up just as you started. You may want to demonstrate what this would look like by walking around the chair.
- Tell the students you can only process one instruction at a time, so their algorithm needs to be step-by-step.
- As students suggest instructions write them on the board or wherever everyone can see them. Their pseudocode will probably end up looking something like this:
    1. Step forward
    2. Turn left
    3. Step forward
    4. Turn left
    5. Step forward
    6. Turn left
    7. Step forward
    8. Turn left



- Go ahead and follow their algorithm to prove that it works. But that's eight lines of code! Tell students that the same instructions can be written using just three lines of code. If they have not noticed already, have students look for places where the code repeats.
- Tell them that whenever you have code that repeats, you have an opportunity to use a loop to simplify your code.
- Prompts:
    - What lines are repeated? *(1) Step forward. (2) Turn left*.
    - How many times are they repeated? Four
    - So how could we rewrite this code? Students will suggest a version of the following: *Repeat 4 times: Step forward, Turn left*
- Go ahead and follow their revised algorithm to prove that it works.

There! They have just rewritten eight lines of code as three lines of code, by using a loop. The 'repeat' command creates a loop. The code within the loop gets repeated a certain number of times until a condition is met. The condition in this algorithm is that the code in the loop is repeated 4 times. Once this condition is met, the program exits the loop.

This is a great opportunity to have the students think of the benefits of having fewer lines of code. *Some possible reasons: Less typing, saves time, fewer chances of making a mistake, easier to read the code, fewer lines of code to debug…*

Notes

- Depending on the particular class, you can make this exercise more challenging, by requiring the students to be more specific in their instructions.

Example: Step forward 14 inches (you can have students actually measure the exact distance), turn left 90 degrees…

05.2 Activity: Loops demos

Microsoft MakeCode has 4 different loop blocks:

- 'Repeat' block
- 'While' block
- 'For' block
- 'Forever' block

To start, the students can code the same algorithm they created in the unplugged activity using a loop.

Activity: 'Repeat' block

The repeat loop is the most basic of loops in MakeCode. I just needs the number of times to repeat the code inside the loop.

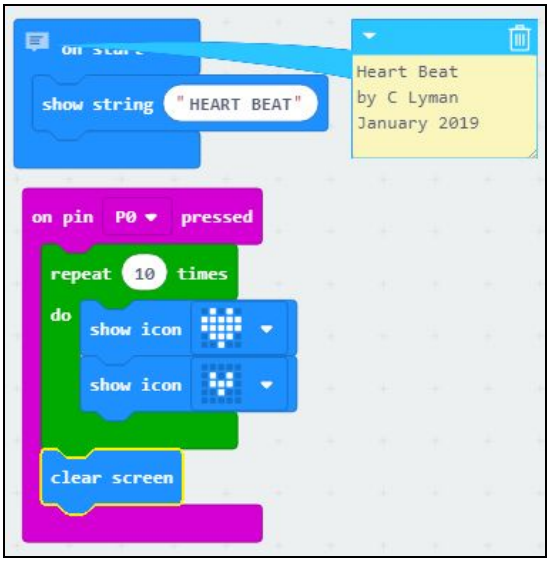
Activity: Heart Beat

The Heart Beat program uses a repeat loop set to a number different than the default 4 and a value up to 10. Instead of using a button to start the loop it use the pin0 pressed which is done by holding the ground with the right thumb and finger and then with the left hand quickly touching the 0 pin and letting it go. The touching with 2 hands makes a connection between the 0 pin and the ground. When the pin0 is connected the program will flash a big heart and a small heart as if the heart is beating. Sometimes it may take several tries to make the pin0 touch work.

In the Simulator the “0” pin can be click with the mouse to test it. Once it is downloaded to the microbit and working, have 2 student hold hands in the middle and one touch the ground and one touch the pin0. In other programs the other pins can be programmed to touch also. This gives the coder 3 additions “buttons” besides “A”, “B”, & “A+B”.

Algorithm and Pseudocode

- *Add comments to the beginning of the program*
- *On pin0 pressed*
  - *Repeat 10 times*
  - *Show icon “Big Heart”*
  - *Show icon “Small Heart”*
- *Clear screen*



Coding Heart Beat MakeCode

The coding of this activity basically follows the algorithm and pseudocode in the plan.

Link to sample Code for Heart Beat:
https://makecode.microbit.org/_KzFAFviArfhk

Activity: Frere Jacques Song

Write music using loops with the song, “Are You Sleeping Brother John?” (Frere Jacques)

https://www.letsplaykidsmusic.com/wp-content/uploads/2015/02/frere-jacques-music.pdf

In the song each line repeats twice so a **Repeat** loop will allow the program to coded more easily. Music plays really pretty good in the Simulator with speakers or headphones.

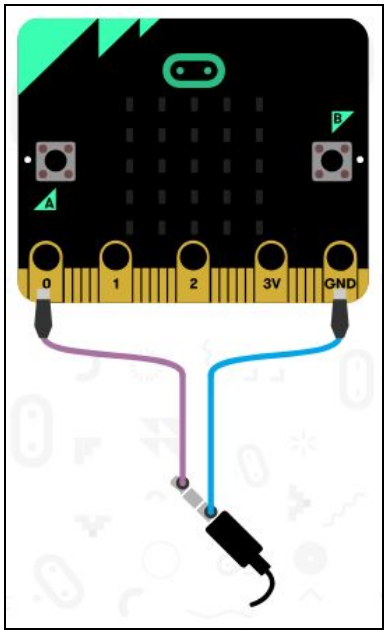
Algorithm & Pseudocode

- *Add comments to the beginning of the program*
- *Adjust tempo as needed*
- *OnButton A pressed*
    - *Repeat 2 times*
        - *Play ¼ notes C D E C*
    - *Repeat 2 times*
        - *Play ¼ notes E F*
        - *Play ½ note G*
    - *Repeat 2 times*
        - *Play ⅛ notes G A G F*
        - *Play ¼ notes E C*
    - *Repeat 2 times*
        - *Play ¼ notes C low G*
        - *Play ½ note C*

Coding Block Code

Link to sample code for Song “Frere Jacques”
https://makecode.microbit.org/_TD1J8F4d5DMW

Connecting headphones to a microbit



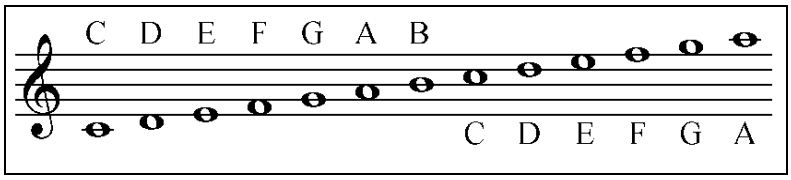
Here is a diagram on how to connect headphones or an external speaker to the microbit. Alligator clips can be used to make the connections. To use sound with your micro:bit, you will need to connect it to some speakers, a buzzer, or headphones. There are several connectors that can be 3D printed from http://thingiverse.com. A small “breadboard” can also be used to make the connections with connecting wires. This diagram shows up in the

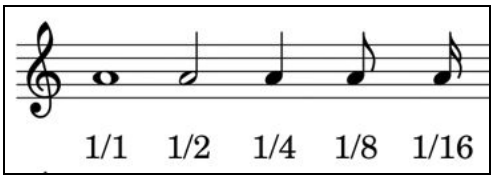
Simulator anytime a Music block is coded in the Workspace.

See how to do this here: Hack you headphones.
https://makecode.microbit.org/projects/hack-your-headphones.

Notes & Note Values

Writing music for a microbit can be written using the **"play tone 'note' for '....' beat"** block of code. When the 'note' is click on a piano key shows up and the name of the note is shown when the mouse is over the note on the piano keyboard.





These activities become good "cross-curricular" activities to help student learn to basically read music. The project at the end of this unit might be one that is coordinated with a music teacher.

"How to Read Music" https://www.musicnotes.com/blog/2014/04/11/how-to-read-sheet-music/

Modifications

- Try using the repeat loop with a "**start melody…**" block. The block has its own repeat at the end of the block.
- Find a song from sheet music and code it
- Write a song of your own

Activity: 'Forever' and 'While' blocks: European siren!

A "forever" loop will run as long as the microbit is on. The "while" loop block is useful when you want your program to loop until a certain event happens or a different condition is met. When one loop is placed inside another loop it is called a "nested" loop. This program will have a nested loop. In this program a "while" loop with a certain condition is set to turn on or turn off the siren loop. A European siren is created by playing 2 notes like middle C and middle F. In this program it will be programmed to turn on when the light is sensed above a certain level. (Light levels varies between 0 'dark' and 255 'bright light'. The microbit uses the LED display as the light sensor.)

The condition in the while loop has to be a Boolean condition which can be evaluated as either "true" or "false". This is the same as the condition in an "if (condition) then …." statement.

Algorithm & Pseudocode

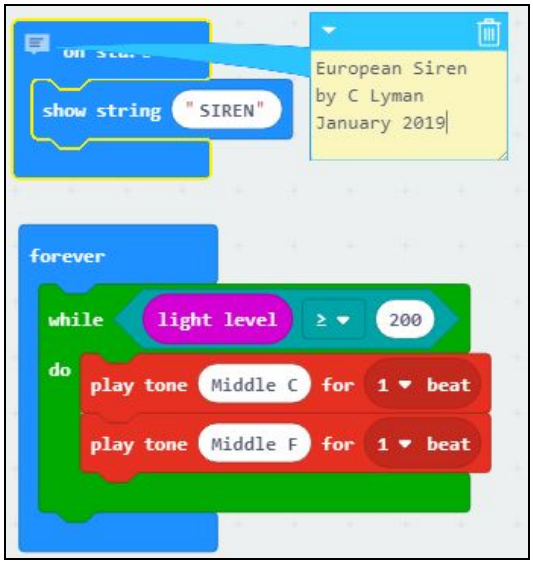
- *Add comments to the beginning of the program*
- *Adjust tempo as needed*

- *In a "forever" loop*
  - *Place a "while" loop with the condition (lightLevel >= 200)*
    - *Play whole note C*
    - *Plot an LED at 2,2*
    - *Play whole note F*
    - *Unplot the LED at 2,2*

Coding block code

Use the algorithm to program the microbit. Test it in the simulator. Download it to a microbit when it is working.

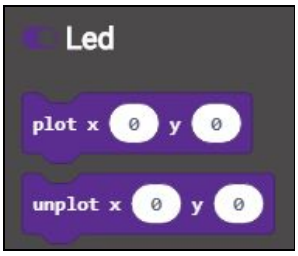
Sample code:
https://makecode.microbit.org/_iwmCfb6U1ECs



Note Modifications

- Try 2 other notes.
- Try a different spread between the notes.
- Change the light level needed to make the while loop work.
- Try another sensor and value to make the siren work.

Turning on a single LED light



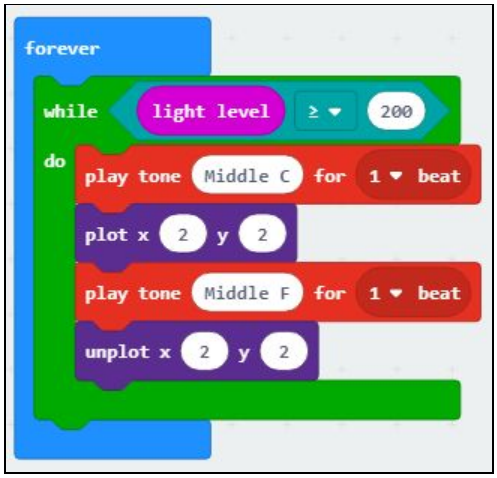
Grid for plotting LEDs				
0,0	1,0	2,0	3,0	4,0
0,1	1,1	2,1	3,1	4,1
0,2	1,2	2,2	3,2	4,2
0,3	1,3	2,3	3,3	4,3
0,4	1,4	2,4	3,4	4,4

To get a "flashing" light on the LED display we will use a command out of the **LED toolbox** to turn on an LED in the 5 by 5 display. The display is 5 by 5 grid with numbers across the top (x-axis) with values 0-4 and the numbers down the side (y-axis) with values 0-4. This would make the top left LED at (0, 0), the center LED at (2,2) and the bottom right LED at (4, 4), LEDs can be turned on using the **plot x (..) y (..)** block and turned off using the **unplot x (..) y (..)** block. To turn the middle LED of the display on it can be done the using the **plot x (2), y (2).** Then turn off the LED using the unplot block. The LED **plot** and **unplot** block can turn LED on and off much quicker than the **show Icon** or the **show LED** blocks. Both of these blocks slow the program down so much that the siren sounds funny.

Plot LED Block Code

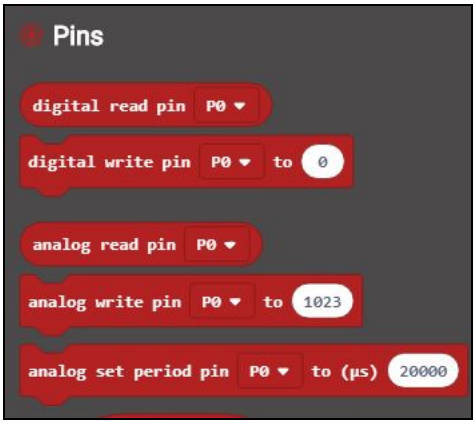
Add the **plot** and **unplot** blocks to turn on the center LED when the siren is running.

Sample code: https://makecode.microbit.org/_2ujE4MP7EdjU

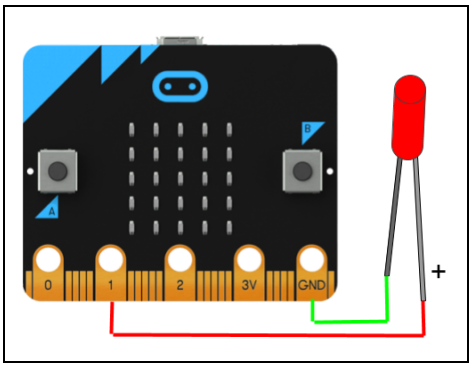


Connecting an external LED

Once the “Siren” is working add an externally connected LED light to pin1 and turn it on and off after each note is played. LEDs have negative ‘**-**’ and a positive ‘**+**’ sides when it is connected it needs to be connected with the ‘+” side to the power (pin0, pin1, or pin2) and the ‘-’ to the ground (GND).



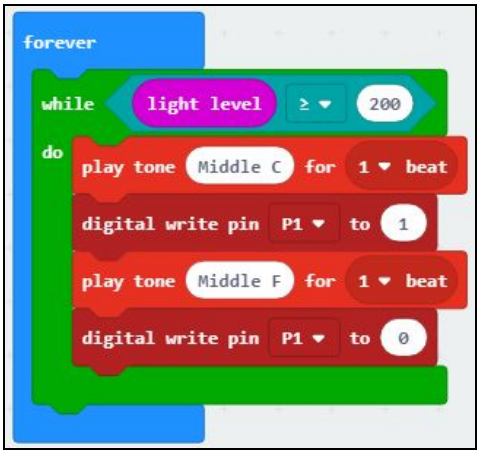
The LED can be powered with block commands from the **Pins toolbox**. The external device, in this case the LED, can be powered with a ‘**digit write pin**’ block with a ‘1’ turning on the power or a ‘0’ turning off the power. A external device can also be powered with a ‘**analog write pin**’ with a value between 0 and 1023. The higher the number the greater the amount of power applied to the external device. This allows the lighting of the LED from a very dim to a bright lighted one.



Just as an LED can be lighted other external devices can be powered with the microbit with up to 3 volts from the power supply to the microbit.

External LED Block Code

Connect an external LED to the microbit at pin1 and the ground with the long wire on the LED to the pin1 and the short wire to the ground. The music is playing to the speaker using pin0 so for this project the external LED is connected to pin1. The value of 1 will light the LED and the value of 0 will



turn off the LED. (The LED and speaker can be connect to different pins.)

Sample code: https://makecode.microbit.org/_35k4s0H5LEtY

Modifications

- Try an “analog write pin (P1) to (value) to light the LED from a dim value at 0 or a bright value at 1023
- Try a different colored LED
- Add a second LED to pin2 that is different color and turn one on then off when the second LED is lit

Activity: ‘For’ block: Counting numbers

The “**for**” loop lets a programming count from a beginning number until an end number reached. The “**for**” loop can also tell what to count by. With the normal count being by 1s. The loop keeps track of what number it is currently on with the variable “index”. The value of “index” increases by 1 each time it loops in a normal loop. Example: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, … last number. The structure of the **for** loop block is shown in the block snapshot. It starts with the key word “**for”** the variable **“index”** from the starting number **“0**” to the end number “**4”**. It counts by 1s as a default but that is not shown in the block.

Historical use of ‘I’ and index

In the programming language, FORTRAN, from the 1950’s “for” loops could only have a single letter variable “I”, “J”, or “K”. Even today many programmers will use a single letter “i” as the variable name for the index of the loop. Programmers also start counting with “0” instead of “1” in most cases. (https://skillcrush.com/2013/01/17/why-programmers-start-counting-at-zero/) It is all about efficiency. If you look at a child’s age, the first year is the zero year. Only after 1 year is a child’s age counted as 1.

Structure of a for loop

In JavaScript the starting, ending number, and count by number can be changed. The variable “index” can also be changed if it needs to be.

```
for (let index = 0; index <= 20; index++) {
  Actions inside the loop
}
```

In the example above the first “index” = 0 is the starting number. The “;” separates the starting number and the phrase the marks the ending number. The ‘ending’ phrase says, “keep doing the loop while index is less than or equal to 20 or whatever the end number is.” The last part of the statement says, “index++”, this means to count by 1 each time it loops. Other ways to write this part of the statement could include:

Counting forward (adding to the index)

- The start number needs to be smaller than the end number and the comparison on the end number needs to say “<=” less than or equal to.
- “index++” is the default and adds one each time to the index
- “index +=1” this is also add one each time to the index
- **“index +=2” this will count by 2s or add 2 to the index each time**

Counting backwards (subtracting from the index)

- The start number needs to be larger than the end number and the comparison on the end number needs to say “>=” greater than or equal to.
- “index--” will subtract 1 for index each time
- “index -=” this will subtract 1 from the index each time
- “index -=2” this will subtract 2 from the index each time

Counting Numbers with ‘for’ loop

In this project the microbit will set up to count numbers using a ‘for’ loop when the onButton A is pressed. It will start by counting from 0 to 10 then the program can be modified to count to another ending number.

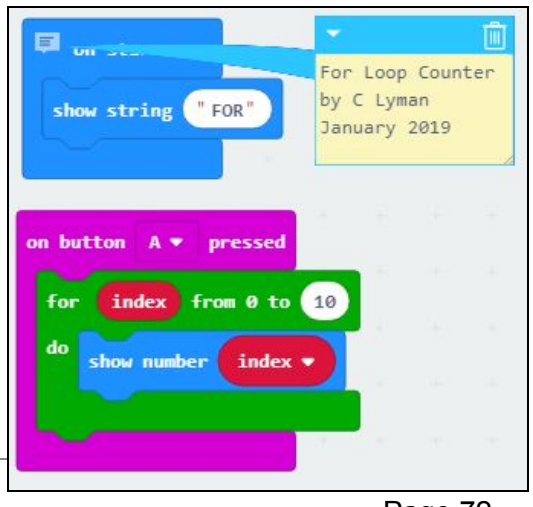
Algorithm & Pseudocode

- *Add comments to the beginning of the program*
- *When the an onButton “A” is pressed start counting*
- *In a ‘for’ loop set the end number to 10*
    - *Place a display current number (index) block inside the loop*

Coding MakeCode for loop

Use the algorithm to program the microbit. Test it in the simulator. Download it to a microbit when it is working.

Link to sample code:
https://makecode.microbit.org/_DTsM4iM16ALi



Modifications

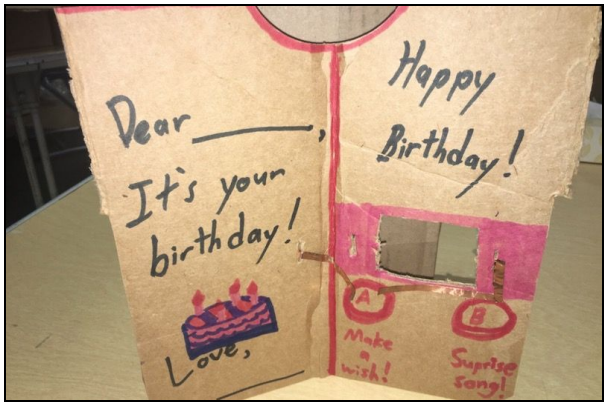
- Make a second loop that counts backwards
- Create a Music Note player loop. Each note has a value on the microbit. Here is a list of the notes by value. Set up a loop that starts with the index at 262 goes to 554. http://bit.ly/microbitnotevalues This will need to be done in the JavaScript version. Blockcode version: https://makecode.microbit.org/_HdJdy0bmK7b3. JavaScript version: https://makecode.microbit.org/_Kka4tvT4FV0L
- Create an **American Siren** by playing notes going up and then down and repeating it in a 'repeat' loop. Also adjust the loop so it counts by 10s up and then down by 10s. https://makecode.microbit.org/_8639u6PxMREJ
- Connect and external LED and use the "analog write pin (P1) to index" inside a 'for' loop from 0 to 1023 so that the LED starts dim and grows brighter. In JavaScript have it count backwards from 1023 to 0 so the LED goes from bright to dim.

05.3 Innovation Project: Loopy Entertainment and Innovation!

There are many different ways to use the three types of loop blocks. By also using the external pins as input sensors and outputs to LEDs or speaker a student can start thinking creatively like and innovator! Enhanced modifications for some of the sample programs could be used

Recall the different common repetitive actions you thought of back at the beginning of this lesson.

- How will you use loops to create something useful, entertaining, or interesting?
- What might you make?



Project Ideas, Design, & Plan

Here are some suggestions:

- Create an animated gif (looping image that changes) and add music that matches.
- Create animation that repeats for one of the melodies included in Make Code (like Happy Birthday).
- Create an electronic greeting, birthday, get well, or other kind of card with music and lights.
- Create different animations that run when different buttons are pressed.
- Make a jukebox player that can play different tunes.
- Create an alarm that includes sound and images. What will set the alarm off? What will make the alarm stop sounding?

- Create a holiday display with music, LED animation and external LEDS.
- Create a piece of “wearable technology” with lights and music.
- Use servo motors to create a creature that dances and changes its expression while a song plays.
- Create a water alarm when 2 wires come in contact with water and then the microbit will play a warning noise and flashing alarm.
- Create a burglar alarm that will play a warning noise and flashing alarm when the microbit is moved or 2 wires come in contact with each other.
-

Example:

Hat Man Project

Hat Man Videos
micro:bit Hat Man
https://youtu.be/Xvybu_T5lL8

micro:bit Hat Man - inside view
https://youtu.be/ZfKgFQjygQQ

This project uses the micro:bit light sensor to display a happy face when it is sunny, and a frowning face when it is dark. The micro:bit is connected to a servo mounted on the inside of the container, and the smile and frown are attached to plastic coffee stirrers with tape and hot glue.

Juke Box

Create a music juke box that will play different tunes using different events for each song.

Loop Modifications

Any of the loop activities could be enhanced using some of the ideas in the modifications.

Reflection

Have students write a reflection of about 150–300 words, addressing the following points:

- Explain how you decided on your particular “loopy” idea. What brainstorming ideas did you come up with?
- What type of loop did you use? For, While, or Repeat
- What was something that was surprising to you about the process of creating this program?
- Describe a difficult point in the process of designing this program, and explain how you resolved it.
- What feedback did your beta testers give you? How did that help you improve your loop demo?

Handout to help writing your reflection. https://goo.gl/tMNa8j or “Coding & Innovation using Microbits” booklet http://bit.ly/codingmicrobitsbooklet

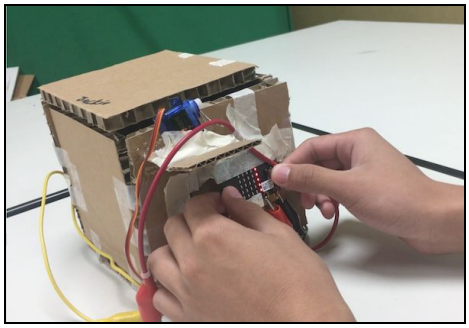
Assessment

Competency scores

Competency	4	3	2	1
Loops	At least 3 different loops are implemented in a meaningful way.	At least 2 loops are implemented in a meaningful way.	At least 1 loop is implemented in a meaningful way.	No variables are implemented.
Variables (parameters)	All variable names are unique and clearly describe what information values the variables hold.	The majority of variable names are unique and clearly describe what information values the variables hold.	Few variable names are unique or clearly describe what information values the variables hold.	None of the variable names clearly describe what information values the variables hold.
Sound, Display, & Motion	Uses sound, display, and motion in a way that is integral to the program.	Uses only two of the required elements in a way that is integral to the program.	Uses only one of the required elements in a way that is integral to the program.	None of the required elements are used.
Micro:bit Program	micro:bit program: 1) Uses loops in a way that is integral to the program 2) Compiles and runs as intended 3) Meaningful comments in code	micro:bit program lacks 1 of the required elements.	micro:bit program lacks 2 of the required elements.	micro:bit program lacks 3 or more of the required elements.
Collaboration Reflection	Reflection piece includes: 1) Brainstorming ideas 2) Construction 3) Programming 4) Beta testing	Reflection piece lacks 1 of the required elements.	Reflection piece lacks 2 of the required elements.	Reflection piece lacks 3 of the required elements.

06 Radio Communications

This lesson covers the use of more than one micro:bit to share and combine data. Students will explore a radio communications and Morse code which was used for many years to send messages over the telegraph and radio. Students will send and receive string messages in a series of guided activities. Student build and send Morse code using numbers and then displaying the number as a 'dot' or 'dash' on the receiving microbit. Finally, students are asked to collaborate so that they can share their micro:bits and create a project together.



Lesson objectives

Students will…

- Understand how to use the Radio blocks to send and receive data between micro:bits
- Understand the specific types of data that can be sent over the Radio

Lesson structure

- Introduction: Radio & communication
- Unplugged Activity: Handwritten Morse Code
- Activity: Send Initials over the radio
- Activity: Send & receive Morse Code over the radio
- Project: Radio
- Assessment: Rubric

Lesson plan

1. **Overview**: Radio and communications
2. **Unplugged**: Handwritten Morse Code
3. **Activity**: Radio Initials and Morse code
4. **Project**: Radio project

Standards — CSTA K-12 Computer Science Standards

- CL.L2-03 Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities.
- CL.L2-04 Exhibit dispositions necessary for collaboration: providing useful feedback, integrating feedback, understanding and accepting multiple perspectives, socialization.
- CL.L2-05 Implement problem solutions using a programming language, including: looping behavior, conditional statements, logic, expressions, variables, and functions.

06.0 Introduction

Radio introduction:

The micro:bit allows you to communicate with other micro:bits in the area using the blocks in the Radio category. You can send a number, a string (a word or series of characters) or a string/number combination in a radio packet. You can also give a micro:bit instructions on what to do when it receives a radio packet.

Kinds of radio communications

Have the student think about all the different kinds of radio communications they can think of. Give them a couple of minutes to write as many as they can in their Reflection booklet. Have them share with a partner add any additional kinds to their list. Have the groups share with the whole class one item at a time from each group.

Possible ideas could include: TV, FM radio, WiFi, cell phones, GPS, satellite, Bluetooth to speakers or headphones, walkie-talkie, police or emergency radios, wireless phones, cell phone controlled sprinklers or security systems, home automation devices, shortwave radio, CB radio, etc.

Vocabulary

transmitter - a radio that can send communications

receiver - a radio that can receive sent communications

medium - the radio waves with which radio communications travel

Wi-Fi - is a wireless local networking technology many used in homes, schools, or businesses

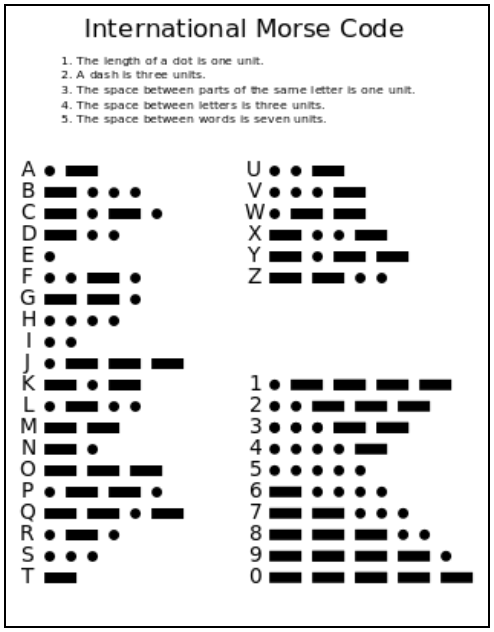
Bluetooth - a wireless technology used to communicate over short distances, mainly between 2 or more devices

Pair Programming:

Note: Many teachers find the concept of “pair programming” to be a valuable way to have students collaborate when programming. Two students share one computer, with one student at the keyboard acting as the driver, and the other student providing directions as the navigator. Students must practice good communication with each other throughout the entire programming process.

06.1 Unplugged: Morse Code

Morse code was one of the first kinds of communication that could be used over a long distance. The first commercial electrical telegraph was developed in May 1837 in London. In the USA Samuel Morse independently developed a telegraph along with the Morse code and sent the first message in January 1838.



At the end of 1894, the young Italian inventor, Guglielmo Marconi, begin working with wireless radio. In March 1897, Marconi transmitted Morse code over a distance of 6 km (3.7 miles). Morse code continued to be used for communication until the 1990s. At one time all Boy Scouts had to learn Morse Code to earn the 1st Class rank.

(Telegraphy. Wikipedia. https://en.wikipedia.org/wiki/Telegraphy)

(Morse Code. Wikipedia. https://en.wikipedia.org/wiki/Morse_code)

Watch the first 1:20 minutes of the 1966 Army training video for Morse code. https://www.youtube.com/watch?v=Li8Hiwbc664 Some student may want to watch the whole video.

One of the most common famous Morse code messages is SOS or ⋅⋅⋅ − − − ⋅⋅⋅ or “di di di dah dah dah di di di” which mean I am in trouble and need help immediately.

Handwritten Morse code activity

Have each student write their first name in Morse Code. This can be done in the Coding & Innovations using Microbits Reflections booklet. Have them see if their partner can read it. Next have them code a short message in Morse code and have their partner decode it.

06.2 Activity: Radio Initials and Morse Code

For radios to work on microbits they need to be on the same channel, just the same as family walkie talkie radios need to be on the same channel to be able to talk to each other. Microbits can have channels up to 255. The first activity will be to set up the microbit to send their initials over the radio and receive any initials to them and display them. The second activity will be to set up microbits to send and receive Morse code.

Activity: Radio Initials

For this project students will work with a partner or in small groups. Divide the students up into groups of 4-6 or with a partner. Have each group choose which channel they will be communicating on. Make sure each group is on a different channel or group number.

In this activity students will send their initials over the radio when the “A” button is pressed. They will also set up their microbit to receive and display any initials they receive in the “onRadio received string” event.

In this activity each student’s microbit will be coded to be a transmitter and a receiver.

Algorithm & Pseudocode

- *Add comments to the beginning of the program*
- *In the onStart event set the channel number for everyone in the group.*
- *In onButton A pressed event*
  - *Use a radio send string with your initials to send a message*
- *In an onRadio received string event*
  - *Display the string when received*

Coding Radio Initials

In the Radio toolbox we will use the following tool blocks: “radio set group (#)”, “radio send string (‘message’)”, and “on radio received (receivedString)” event.

Comments & radio group

In the onStart event add a title, coder’s name, and the date. From the Radio toolbox bring a **“radio set group (#)**” block and set the channel number for the group.

Message sender

In an “onButton ‘A’ pressed” event add a “**radio send string (‘..’)**” block. Inside the message quotes add you first and last initials in all CAPS.

Message receiver

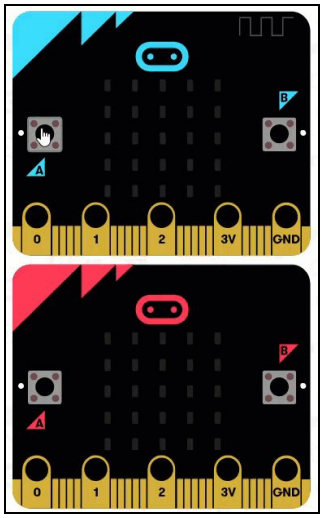
From the Radio toolbox add an “**on radio received (receivedString**)” event. In this block it comes with a predefined variable **receivedString.**

Since receivedString is a variable that will hold the value of the received message it is not in quotes. In the event block bring a “show string (‘...’)” block. From the Variables toolbox bring in a **receivedString** varible in place of the quotes in the block. This will take whatever string is received and display it on the microbit.

Sample block code program: https://makecode.microbit.org/_VwzFWJFiKXw3

Testing in Simulator

In the Simulator when the button ‘A’ is pressed to send the initials message over the radio a second microbit shows up in the Simulator. When the message is sent again for the first microbit it shows up on the second microbit. If the ‘A’ button is pressed on the second microbit it sends the message back to the first microbit in the Simulator. This is really awesome to be able to test the program in 2 microbits and never having to download it.



Download and test

Download the program to the microbit and test it with others in the group. If anyone in the group cannot see the others initials have them check to make sure they are on the same radio group. If possible have the student go out into the hallway or outside to see how far away they can send and receive the initials message.

Modifications

- Have the students program another message to be sent when ‘B’ button is pressed
- Using a “**radio send number (#)”** block and an “**on number received event”.** Try sending the temperature from a remote microbit to another microbit.
- Explore projects in the MakeCode Microbit course “***Science Experiments***” for additional projects using radio communications. https://makecode.microbit.org/courses/ucp-science This can lead to many different cross-curricular activities with a science teacher.

Activity: Morse Code

For this Morse Code activity the “sender” will use an ‘A’ button to send a number ‘0’ and the ‘B’ button will be used to send a number ‘1’. The “receiving” microbit will receive the number and then then convert it into a “dot’ or a “dash” on the LED display. Student can refer to the Morse Code table in their “Coding & Innovation using Microbits Reflection” booklet.

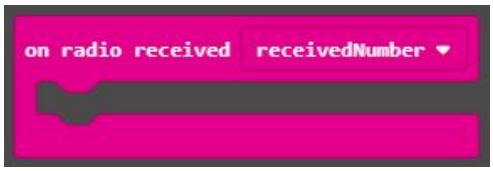
In this activity each student’s microbit will be coded to be a transmitter and a receiver.

Algorithm & Pseudocode

- Add comments to the beginning of the program
- In the onStart event set the channel number for everyone in the group.
- In onButton A pressed event
    - Use a radio send number to send the number 0 for a “dot”
- In onButton B pressed event
    - Use a radio send number to send the number 1 for a “dash”
    - string with your initials to send a message
- In an onRadio received number event
    - Use an if blcok of code to check to see if the number is a 0 or a 1
    - If the (receivedNumber) is 0 then
        - display “dot” icon on the screen
        - Pause for a part of a second
        - Clear the screen
    - Else
        - Display a”dash” on the screen
        - Pause for a part of a second
        - Clear the screen

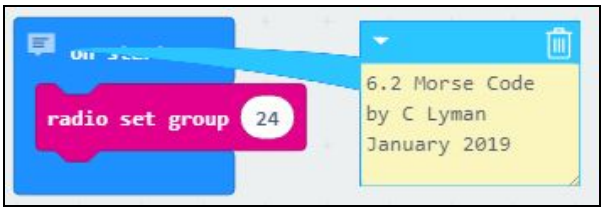
Coding Morse Code

In the Radio toolbox we will use the following tool blocks: “radio set group (#)”, “radio send number (#)”, and “on radio received (receivedNumber)” event.



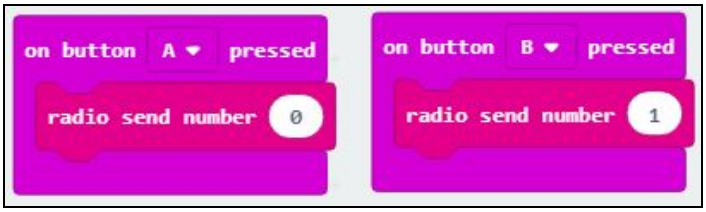
Comments & radio group

In the onStart event add a title, coder’s name, and the date. From the Radio toolbox bring a **“radio set group (#)**” block and set the channel number for the group.



Message senders

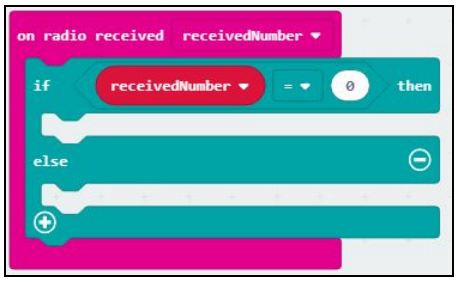
In an “onButton ‘A’ pressed” event add a “**radio send number (0)**” block. Inside the parentheses add a ‘0’ for the message to be sent for a “dot”.



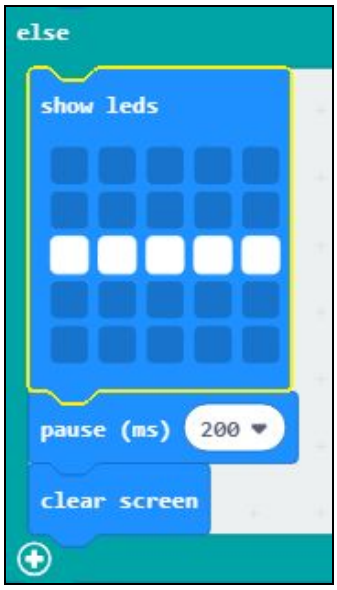
In an “onButton ‘B’ pressed” event add a “**radio send number (1)**” block. Inside the parentheses add a ‘1’ for the message to be sent for a “dash”.

Message receiver

From the Radio toolbox add an “**on radio received (receivedNumber**)” event. In this block it comes with a predefined variable **receivedNumber.** Since receivedNumber is a variable that will hold the value of the received number.



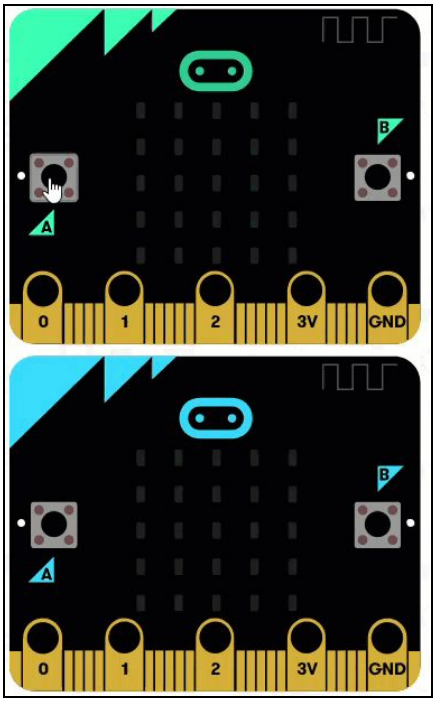
From the Control toolbox add an “if (...) then … else …” block. In the condition add a comparison block to see if “**(receivedNumber == 0)”.** In the block following the “**then”** and block to “**show icon ‘dot’**”, “**pause ms (200)**”, and a **“clear screen**” block. Following the “else” part of the decision structure add a “**show leds”** block with a dash in the middle followed by a “**pause ms (200)**”, and a **“clear screen**” block.



Sample block code program:
https://makecode.microbit.org/_TK68qqabJfCs

Testing in Simulator

In the Simulator when the button 'A' is pressed to sends a "dot". When the 'B' button is pressed it sends a "dash". The person sending pauses briefly between letters. The sent message travels over the radio and shows up on the second microbit in the Simulator. The second microbit can send Morse Code back to the first microbit. Create a message and test it in the Simulator.



Download and test

Download the program to the microbit and test it with others in the group. If anyone in the group cannot see the "dots" and "dashes" have them check to make sure they are on the same radio group. Have the students compose a message in Morse Code and send it to others in the group. Students may need to write down the "dots" and "dashes" and then use the Morse Code table to translate its. If possible have the student go out into the hallway or outside to see how far away they can send and receive the initials message.

Modifications

- Have the students add a short music tone for a "dot" and a long music tone for a "dash"
- Have students see how fast they can send a message and have it read on the other end

06.3 Innovation Project: Radio Project

For this project, students should work in pairs to design a project that incorporates radio communication to send and receive data in some way. Some projects may have two separate programs: One that receives data, and one that sends data. Students might each choose to submit one program in that case.

Project Ideas, Design, & Plan

In other cases, a pair of students might submit one program that has both sending and receiving code in it, and the same code is uploaded to two or more micro:bits.

Project Ideas

Stop, thief!

Design an alarm system for your bedroom that alerts you with a screen animation when someone opens your door. You can mount one micro:bit on your door and use the accelerometer to send a signal over the radio when it is being moved.

Science Remote Data Collection

Ideas for science projects and radios can be found at:
https://makecode.microbit.org/courses/ucp-science

Weather Station

Create a remote weather station that report temperature, light level, etc. and reports it to another microbit over the radio.

Remote Control Holiday Decoration

Create a holiday decoration that uses music and lights that is controlled remotely from another microbit.

Basement Water Alarm

Ideas for a water alarm can be found at:
https://sites.google.com/view/utahcodingproject/microbits/challenges or
https://drive.google.com/open?id=1suiOlQjuIPSeSnt2sWNjlngcVmxdKTrmaxupk9sVnbc

Text Message Radios

Set up to microbit so they can create and send text messages. Suggestions can be found at:
https://makecode.microbit.org/_h7ci6s9mbisF

Reflection

Have students write a reflection of about 150–300 words, addressing the following points:

- What kind of Project did you do? How did you decide what to pick?
- How does your project use radio communication?
- Are there separate programs for the Sender and the Receiver micro:bits? Or 1 program for both?
- Describe something in your project that you are proud of.
- Describe a difficult point in the process of designing this program, and explain how you resolved it.
- What feedback did your beta testers give you? How did that help you improve your design?

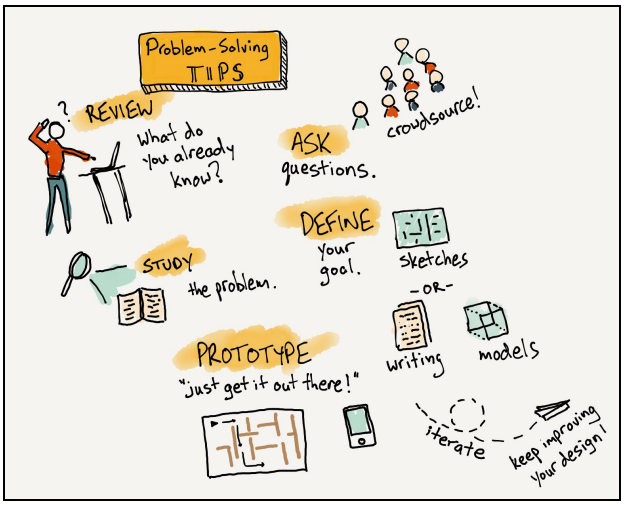
Handout for help with written reflection. https://goo.gl/34pxsj or “Coding & Innovation using Microbits” booklet http://bit.ly/codingmicrobitsbooklet.

Assessment

Competency scores

Competency	4	3	2	1
Radio	Effectively uses the Radio to send and receive data, with meaningful actions and responses for each.	Effectively uses the Radio to send or receive data, with meaningful actions and responses for each.	Use of Radio is incomplete or non-functional and/or tangential to operation of program.	No working and/or meaningful use of Radio.
Micro:bit Program	micro:bit program: 1) Uses Radio blocks in a way that is integral to the program 2) Compiles and runs as intended 3) Meaningful comments in code	micro:bit program lacks 1 of the required elements.	micro:bit program lacks 2 of the required elements.	micro:bit program lacks all of the required elements.
Collaboration Reflection	Reflection piece addresses all prompts.	Reflection piece lacks 1 of the required elements.	Reflection piece lacks 2 of the required elements.	Reflection piece lacks 3 of the required elements

07 Innovation Mini-Project



In this unit, we will be reviewing the concepts we covered in the previous weeks, and providing some ideas for an independent "mini-project" students can focus on in the next several classes. We will also introduce a framework for keeping students accountable to the work they are doing individually and in groups, and providing a rubric for assessment of the development process, as well as the finished product.

It is important to allow students to practice accounting for the work they are doing on a short "mini-project" like this, so that when they move on to an independent project spanning multiple weeks, it will be easier for you to keep track of what everybody is doing.

It also reinforces the important idea that how you solve problems is at least as important to learning as whether you solved them at all (or even got the right answer). Programming is a process of patient problem-solving, and finding ways to value, acknowledge, and reward the problem-solving process is an important part of assessment.

Lesson plan

1. **Review**: Looking back at what we've learned so far
2. **Activity**: Collaboratively independent
3. **Project**: Mini-project

Standards — CSTA K-12 Computer Science Standards

- CL.L2-03 Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities.
- CL.L2-04 Exhibit dispositions necessary for collaboration: providing useful feedback, integrating feedback, understanding and accepting multiple perspectives, socialization.
- CL.L2-05 Implement problem solutions using a programming language, including: looping behavior, conditional statements, logic, expressions, variables, and functions.

07.1 Review

Take this time to review the concepts we have covered so far.

Making

The micro:bit is very effective at bringing real things to life. It can be supported in a cardboard holder, attached to a wand, or even sewn into fabric. The design thinking process is a helpful way to gather more information about the person who will be using whatever you are designing.

Processing and algorithms

The code you write for the micro:bit processes data from its inputs, and outputs it in some way. An algorithm is a series of specific instructions, or steps, that solve a problem or accomplish a task.

Variables

Variables store information so that it can be accessed or referenced later. Some variables hold information that changes, and some hold information that stays constant. It is important to name your variables with something that explains what type of information it holds. Using variables in your code allows you to create algorithms that use mathematical operations to perform the same calculations every time, even when the values of your variables are different.

Conditionals

Conditional statements tell the computer when to do something. They are used to create branches, or decision points, where a program can choose one path or the other based on the values of certain variables, or based on data from the microbit's inputs. Conditional statements can be nested inside one another so that both conditions must be true in order for the enclosed statements to run.

Looping and iteration

Portions of your code can be made to run over and over by using a Repeat or a For block loop. This allows you to loop over several different variables, or items in a group, and do something to each of them. You can also combine a conditional statement a conditional statement and a loop by using a while block, which will repeat until a certain condition becomes true.

Radio Communications

Two microbits can be setup to send numbers or text to each other.

07.2 Activity: Collaboratively independent

Teachers want their students to collaborate on projects but they also want to be able to hold them accountable for getting their work done. Many teachers struggle with assessing exactly how much each individual contributed to a group project, as well as making sure that everyone does his or her "fair share".

The Mini-Project (and the Final Project) are not group projects. Students are asked to propose their own independent project and are expected to get it done. But they are not on their own in this process! We build in frequent opportunities for students to collaborate and share the collective knowledge of the class as they go. We ask them to be "collaboratively independent."

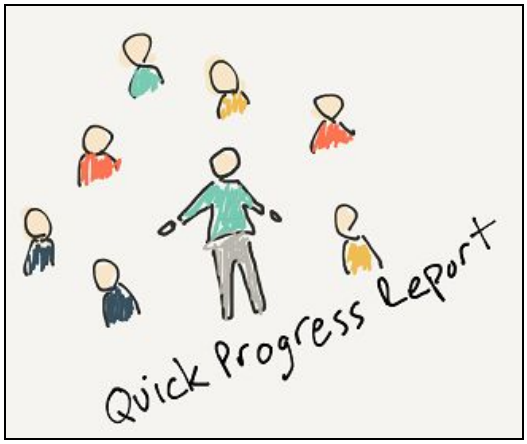
Scrum project management concepts could be introduced at this point. Here are some links to introduce Scrum to students.

- Lesson plan to teach Teamwork & Scrum. https://goo.gl/tEv2qD
- Scrum Method guidelines handout. https://goo.gl/4CiHYM

Here is how we structure our classes:

Beginning of class

For groups of 15 or so, have students each day ***briefly*** (no more than 30 seconds or so) report on their progress in front of the group:



- One-line description of project.
- Their progress so far. What they did yesterday.
- Something they are going to work on today.
- Any difficulties or problems they need help with.

It is important that everyone else is listening to each project and volunteering their help or solutions if they are figuring out the same thing or if they have solved that problem in a previous class.

Example: *I'm working on a pinball machine. So far I have done the board and the ramp. Today I am going to be working on wiring the bumpers so that when the ball hits the bumper, the micro:bit detects it and displays the score.*

Sample response from a classmate: *Yesterday I wired up my targets so that when you throw a ball it keeps score. I can show you how I did it.*

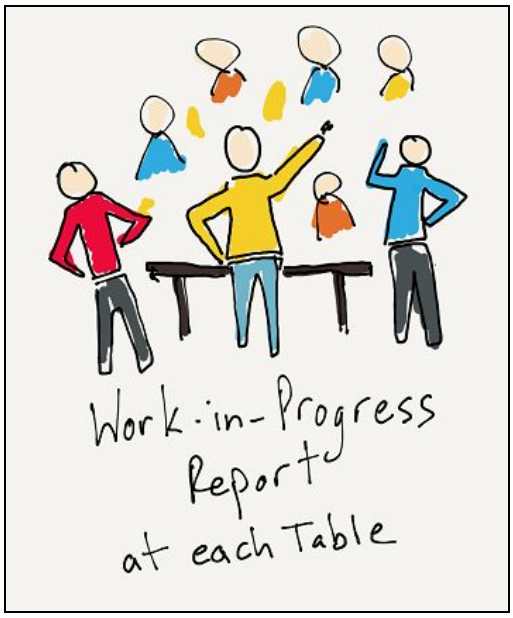
Ideally students who are working on projects should be aware of what other students are working on and what they are figuring out. It creates more opportunities for collaboration in the classroom

and can encourage students to seek help from each other rather than all waiting in line to talk to you.

It's important to hear from everybody but it shouldn't take more than five or ten minutes. For groups larger than 15 or 20 students, you may want to split them into two or three larger groups and have them report out to each other.

During class

This is a time to circulate and check in with students individually, starting with those students who seem to still be stuck from last time. For the most part, students should be working on their projects in small groups, helping each other wherever and whenever possible.



End of class

About ten minutes before the end of class, you can have students do a "work-in-progress" report. Gather the students together and have them move from table to table while each student presents one thing that he or she figured out during the class. This is really an informal presentation, and it is understood that it is not finished at all; it is still a "work in progress." But everyone needs to show something, and the entire group needs to move as one throughout the classroom, almost like physicians making rounds in a hospital. This is an important way to spread ideas throughout the classroom, and to "cross-pollinate" with helpful tips and techniques.

Work-in-progress reports should be short, no more than twenty or thirty seconds. If you have a large class, you might divide the class into several large groups and have them present to each other.

07.3 Innovation Project: Mini-Project

This project takes approximately a week to complete. Most of that time is spent working on the project in a makerspace or art classroom.

Project Ideas, Design, & Plan

The mini-project is an opportunity for students to design a project that serves a purpose by solving a problem or filling a need. It is also an opportunity to do two things:

- Show what you know
- Learn something new

Ideally, there should be a maker component to this project. This is a real world component that works with the code on the micro:bit to do something unique.

Students are asked to each propose an original independent project. Students are allowed to work on the same idea, but they cannot turn in the same code. They can, and should work collaboratively, solving the same kinds of problems together, but the projects they turn in should be unique and original.

Showcasing student work

Students will be showing their work regularly to each other in informal ways. Think about also organizing a day or an evening when parents, administrators, or others from the community are invited to come and view the students' projects. When students can "publish" their work for others their work starts to have real value.

We find that a "science fair" type of setup works well here, with students stationed at their own tables, showing off and demonstrating their project. An event like this works well for these reasons:

- A real world audience for the work students have done can be very motivating
- It is a chance for people who are not familiar with the micro:bit to appreciate the finished product
- It provides good feedback to students about how someone interacts with their product
- It is a chance to have real conversations with the people behind the product, rather than just viewing the product on display by itself
- Finally, and most importantly, it is a chance to bring the community together to celebrate the great work all of your students have done!

Assignment

- Create an original project using the micro:bit.
- Incorporate a physical component to the project.
- Demonstrate the use of one of the following concepts:
- Input / Processing / Output
- Variables
- Simple Circuits
- Iteration/Loops
- Conditional Statements

Project ideas

- Make a "New and Improved" Fidget Cube
- Make a Moving Monster
- Make a musical instrument
- Fishing Game
- Make an Air Guitar (uses while loop to do tempo and pitch)
- Screensaver
- Screensaver that uses other inputs to draw

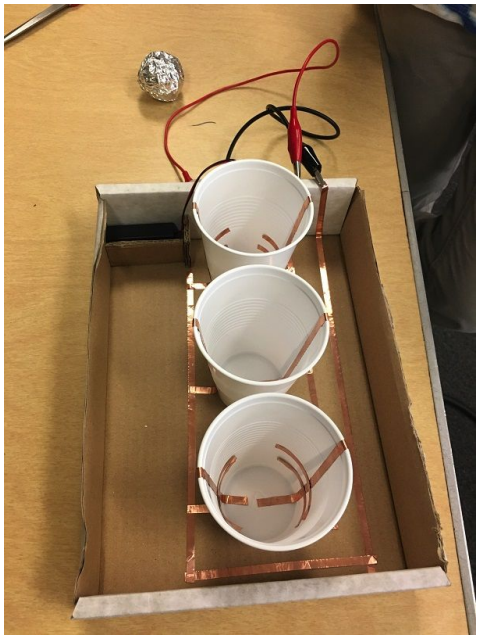
- Interactive book
- Binary Clock or some other way to represent numbers visually

View projects at the following sites for inspiration:

- http://make.techwillsaveus.com/bbc-microbit
- http://microbit.org/ideas/
- https://twitter.com/MicroMonstersUK
- Projects -- https://makecode.microbit.org/projects

Examples

Toss the Ball



This is a skill game in which an aluminum foil ball is thrown into a plastic cup. Copper tape lining the sides bottom of the cup completes the circuit when the ball touches it.

micro:bit Bullseye Project

https://youtu.be/NZUpoSixf4E

This is a skill game in which tennis balls are thrown underhand at one of the three rings, which are lined with aluminum foil so they complete a circuit underneath when the ball makes contact with the ring.

micro:bit Storybook

https://youtu.be/yg1NNLMqa9c

This is a prototype of a storybook that could use the micro:bit to display animations for part of the story. Copper tape is used on the underside of the paper flaps to make contact between the GND pin and each of the other pins in sequence.

Work logs

Because students are working on the projects in class, and much of the benefit comes from working together to solve problems, they should account for the work they are doing by writing a work log.

A work log is a short, bullet point list of what they worked on, and how long it took. Stick to the facts. It shouldn't take more than thirty seconds or so to write up a work log. Students should do one for every class. A shared Microsoft OneNote notebook is a great way to keep a work log that students can update regularly. Alternately, you might use a collaborative shared document, or your classroom management system, or even e-mail.

Sample Work Log

April 11

20 min. Created code that reacts when pins P0 and P1 are pressed.
0 min. Talked with Mr. Kiang about how to attach wires so they won't fall off
20 min. Put target back together with pins
10 min. Helped Cody with attaching his scoreboard

Handout for recording daily Work Log. https://goo.gl/JHZYWn

Reflection

At the end of the week, students should compose a final reflection that summarizes the process of their learning over the course of the week. They should go back through their work logs and talk about the following:

- Talk about one challenge you faced in creating this project, either a challenge in coding or in making the artifact. How did you overcome this challenge?
- What did you demonstrate that you already knew?
- What was the new thing you learned in order to make this? How did you learn about it?
- Who in the class provided help to you along the way? How?
- Describe one specific thing you are proud of in this project.
- What would you do differently next time?
- If you had another week to work on this project, what might you add or improve?

Sample Reflection (excerpt)

"I spent this week finishing up little details with my program, making it work better and more user friendly. The part that surprised me the most was the little things that kept popping into my head, little suggestions that could potentially be good to add, but might not be necessary or even useful. At the beginning of the assignment, I just added them as quickly as I thought of them, but as the project neared the midpoint and conclusion, I find myself considering if I actually need them (as previous additions have been since quickly deleted). Another thing that I find interesting about this is that it is a rather specialized project. Not many people would use it except for me. However, this is supposed to be easily used by other people, so I have to take them into consideration as I design the project. I also realized that I had, at some point, broken part of my code without realizing it, so I now have to fix part of it. The reason that it is a problem is because I added a lot of code at once without deleting it, which is unfortunate. Next time I will add small amounts of code and test it first."

Handout to help write the project reflections. https://goo.gl/FMKYGZ

Assessment

Competency scores

Competency	4	3	2	1
Code - Show what you know	Code very effectively demonstrates the use of previous concept(s). Variable names are unique and clearly describe what information values the variables hold. Code is highly efficient. Code is commented.	Code only partially demonstrates previous concepts, and/or is not efficient.	Code only partially demonstrates previous concepts, and/or is not efficient, variable names not clear.	Code does not demonstrate previous concepts, is not efficient, variable names not clear.
Code - Show something new	Code very effectively demonstrates the use of new concept(s). Variable names are unique and clearly describe what information values the variables hold. Code is highly efficient. Code is commented.	Code only minimally demonstrates new concepts, and/or is not efficient.	Code only minimally demonstrates new concepts, and/or is not efficient, variable names not clear.	Code does not demonstrate new concepts, is not efficient, variable names not clear.
Maker Component	Tangible component is tightly integrated with the micro:bit and each relies heavily on the other to make the project complete.	Tangible component is somewhat integrated with the micro:bit but is not essential.	Tangible component does not add to the functionality of the program.	No tangible component.
Work Logs	All work logs submitted on time, and accurate.	One late or missing work log and/or work logs not accurate nor sufficiently detailed.	Two late or missing work logs and/or work logs not accurate nor sufficiently detailed.	More than two late or missing work logs and/or not accurate nor sufficiently detailed.
Reflection	Reflection piece describes: 1) Development Process 2) Something new 3) Something proud of 4) Future modifications	Reflection piece lacks 1 of the required elements.	Reflection piece lacks 2 of the required elements.	Reflection piece lacks 3 of the required elements.

References

Using the WebUSB interface.

Beta testing - Web USB

Modified on: Wed, 26 Sep, 2018 at 7:27 PM

The Web USB API facilitates communicating with USB devices from the Browser. The API is currently available in **Chrome** and is supported in **beta releases** of the Javascript Block Editor and the Python Editor in Chrome v65+. This enables you to flash your micro:bit straight from the browser without the need to save the .hex file first, and use serial communication between the micro:bit and the editor.

Setup

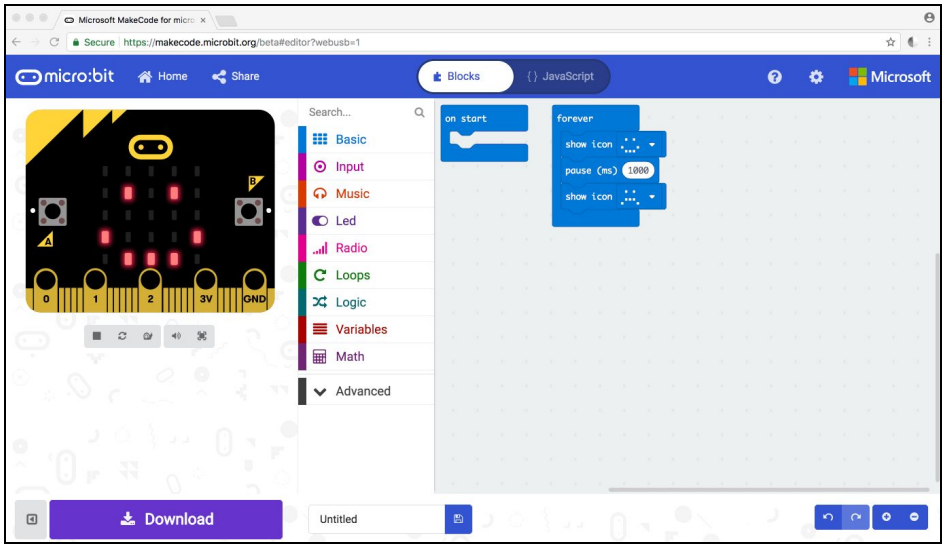
1. Using Windows 8+, Mac or Linux
2. Open this article in **Google Chrome v 65+** (other browsers do not yet have Web USB support) and download the latest version of DAPLink (**0250**) attached to the bottom of the article,
3. Update the firmware on your micro:bit to **0250** by following our usual instructions for performing a firmware update, replacing 0243 with **0250**. If the update is successful, you should see 'WebUSB' listed in the supported protocols of the DETAILS.TXT on the MICROBIT drive.
4.
```
# DAPLink Firmware - see https://mbed.com/daplink
Unique ID: 9901000049624e45005b400f000000250000000097969901
HIC ID: 97969901
Auto Reset: 1
Automation allowed: 0
Overflow detection: 0
Daplink Mode: Interface
Interface Version: 0250
Bootloader Version: 0246
Git SHA: 682d8303e37355532402b8d93c4f240a3cec02a9
Local Mods: 0
USB Interfaces: MSD, CDC, HID, WebUSB
Bootloader CRC: 0xf9d354f5
Interface CRC: 0x3f2b7e12
Remount count: 0
URL: https://microbit.org/device/?id=9901&v=0250
```
5. Your micro:bit is now ready to test with WebUSB.

If you experience a **device not found** issue on Windows, check that the mBed driver is installed properly.

Testing Web USB in the Javascript Blocks Editor

1. Open https://makecode.microbit.org/beta#editor which is a beta instance of the makecode editor with Web USB enabled.

2. The settings menu in makecode should contain an additional menu item entitled **Pair Device,** select this and follow the browser instructions to select your device and **connect**.You will see a notification in makecode that your device is paired and asking you to try downloading now.



3. Try Downloading! You should see that the micro:bit being flashed instantly. The first time you do this, it may take a bit longer, but subsequent flashes should be faster than when using drag-and-drop.
4. For issues with MakeCode and webUSB, please open a support ticket or if you are comfortable using Github, file an issue onthe PXT repository. (https://github.com/Microsoft/pxt-microbit/issues)

Making Student Booklets - Google Docs & Adobe Acrobat

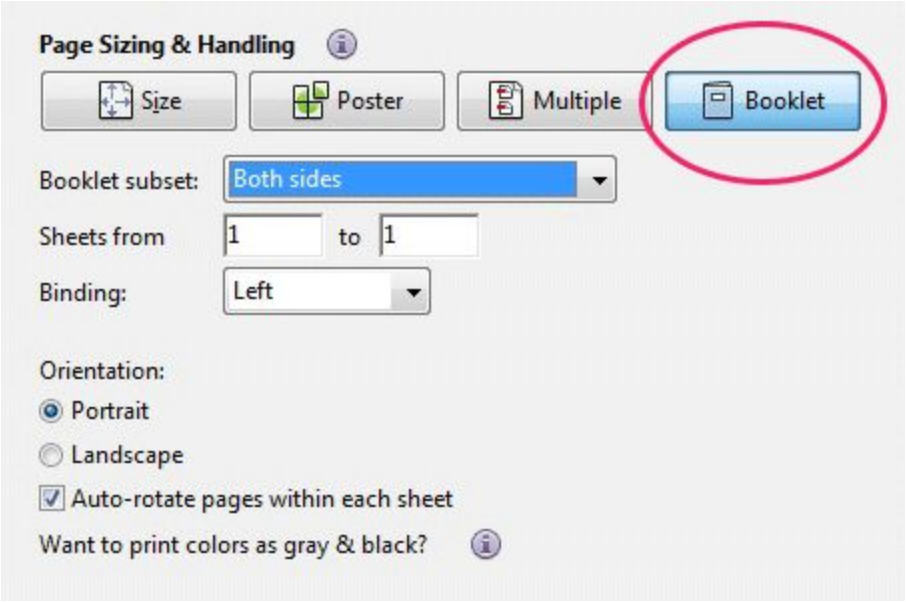
How to make a booklet in Google Docs:

1. Set page size 8.5 x 14
2. Create document so it has multiple of 4 number of pages
3. Set page numbering to start on 2nd page
4. Set margins
    a. Top 1.0
    b. Bottom 0.75
    c. Right 0.75
    d. Left 0.75
5. Set text sizes to 12-18
6. Download as PDF
7. Print as directed below

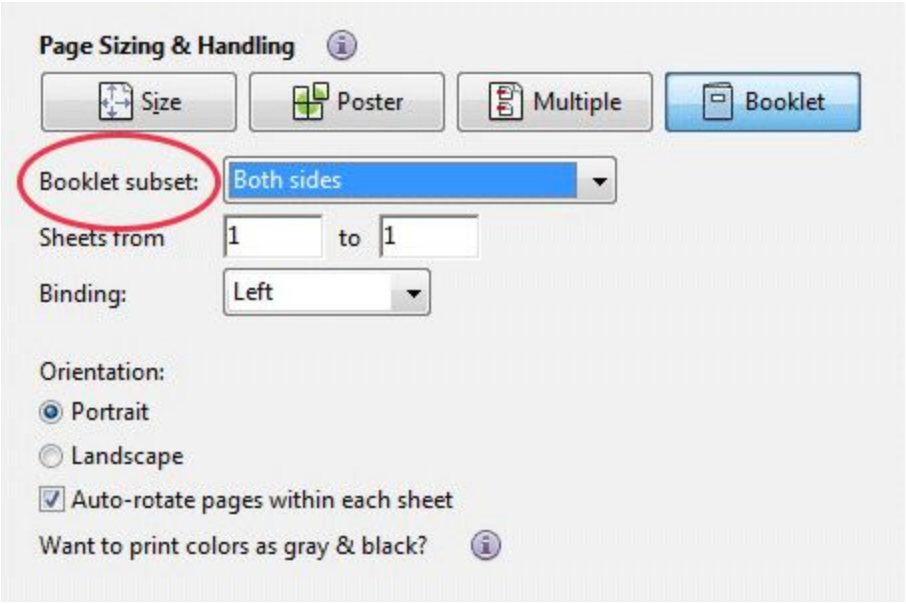
Reader and Acrobat X

Print a multi-page document as booklet:

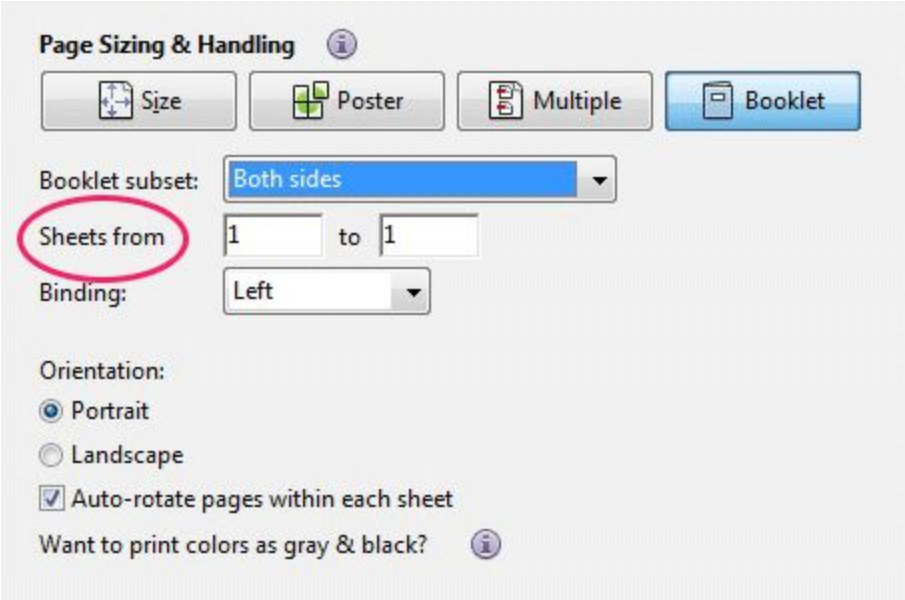
Choose File > Print.
Select a printer from the menu at the top of the Print dialog box.
In the Pages to Print area, select which pages you want in the booklet.
All prints pages from front to back.
Pages specifies a page range for printing a smaller grouping of a large booklet.
You divide a large booklet into smaller groupings, and then print each page range separately.
Under Page Sizing & Handling, choose Booklet.



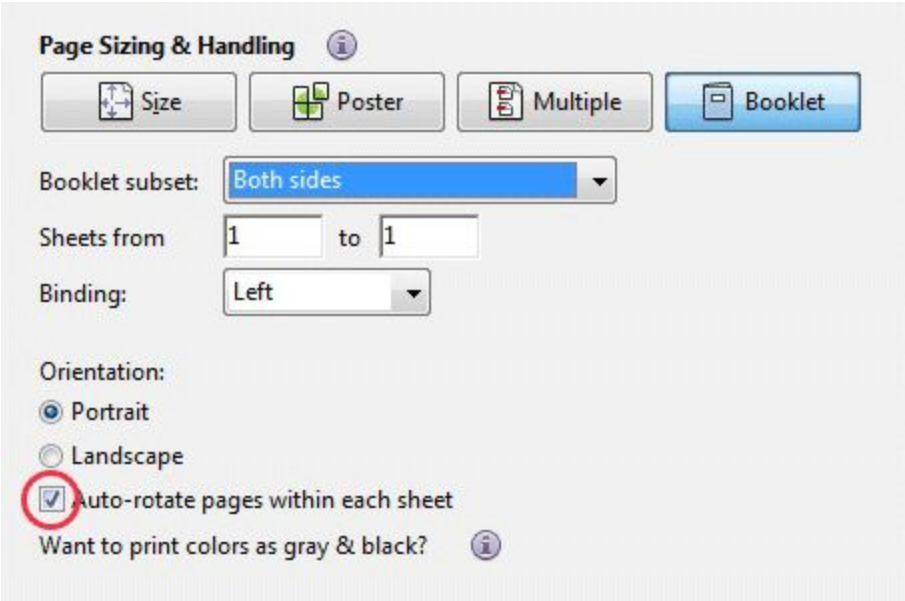
In the Booklet Subset pop-up menu, select one of the following options: Both sides(duplex printers) automatically prints both sides of the paper, if your printer supports automatic duplex printing, or Front side only / Back side only (for non-duplex printers). If your printer can't automatically print both sides, you can first print the front sides of the paper. Then reload those pages and print the back sides.

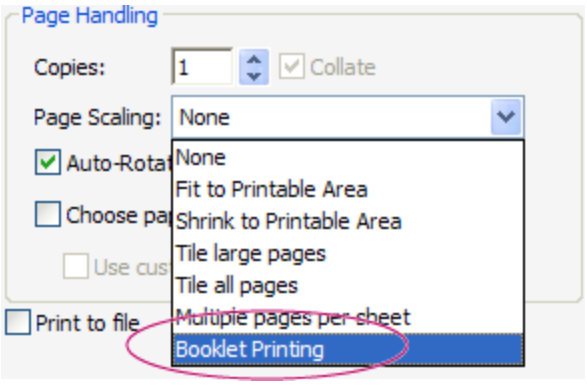


Leave the numbers in the Sheets From boxes as they are. Acrobat or Reader determines which sheets must print to accommodate the print job. For example, if you have a 16-page PDF and you selected All in the Print Range area, then sheets 1 through 4 print.

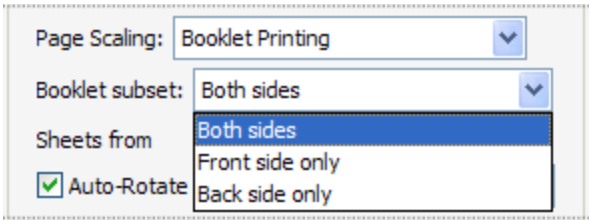


Select Auto-Rotate Pages to automatically rotate each page for the best fit in the printable area.

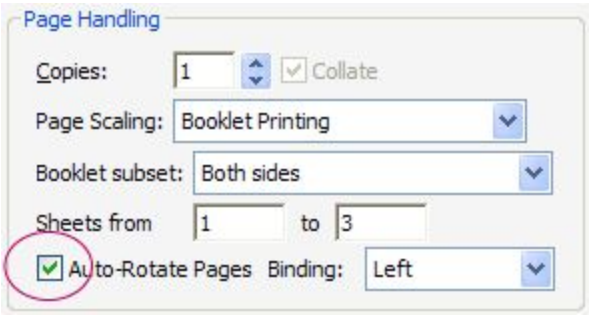




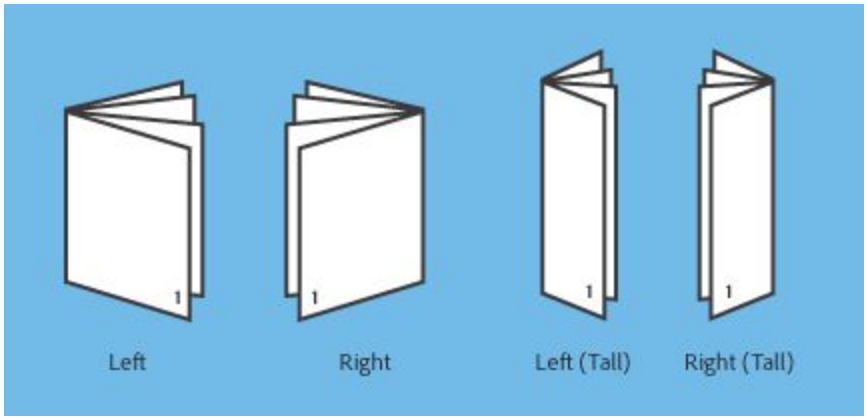
In the Booklet Subset pop-up menu, select one of the following options:
Both sides (Duplex printers) Automatically prints both sides of the paper, if your printer supports automatic duplex printing.



Front side only / Back side only (for non-duplex printers) If your printer can't automatically print both sides, you can first print the front sides of the paper. Then reload those pages and print the back sides.
Leave the numbers in the Sheets From boxes as they are. Acrobat or Reader determines which sheets must print to accommodate the print job. For example, if you have a 16-page PDF and you selected All in the Print Range area, then sheets 1 through 4 print.
Select Auto-Rotate Pages to automatically rotate each page for the best fit in the printable area.



Choose an option from the Binding pop-up menu:



Click OK or Print

Table of Contents